TROPICAL RAINFALL MEASURING MISSION SCIENCE DATA AND INFORMATION SYSTEM

Interface Control Specification Between the Tropical Rainfall Measuring Mission Science Data and Information System (TSDIS) and the TSDIS Science User (TSU)

TSDIS-P907

Volume 2:
Programmer's Guide

Release 3/Draft

Prepared for:

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
GODDARD SPACE FLIGHT CENTER
Code 902
Greenbelt, Maryland 20771

Prepared by:

GENERAL SCIENCES CORPORATION
Laurel, Maryland 20707
Contract Number - NAS5-32351

July 1, 1996

TABLE OF CONTENTS

1. INTRODUCTION

The purpose of the Toolkit is twofold. First, the Toolkit provides a set of commonly used routines, constants, and macros for Algorithm Developers. These commonly used items have been placed in the Toolkit to reduce the amount of parallel code development by the Algorithm Developers. For example, this will mitigate the need for each Algorithm Developer to code his or her own I/O routines.

The routines are designed to be easily used by the Algorithm Developers at their home institutions. This means that the routines contain basic functionality that will be used by most Algorithm Developers.

The second purpose of the Toolkit is to allow seamless integration of TRMM algorithms into the TSDIS environment. Since TSDIS treats the delivered algorithms as black boxes, it is essential that the interfaces with TSDIS be well defined and consistent across Algorithms. Thus, the Toolkit development has concentrated on those routines that are essential to the interaction with the TSDIS environment.

In Section 2 we describe the categories of Toolkit routines being developed, the routines that are found in each category, and a general outline of how the routines can be used together. This is followed in Section 3 by a description of each routine, along with simple examples of how the routines are used. The Parameter Dictionary, which defines each of the parameters in the calling sequence of each routine, is where the developer should look for details of each parameter.

The toolkit routines were designed and developed based on file specifications contained in release 2 of the ICS, volume 3 (Level 1 file specs) and volume 4 (Level 2 and 3 file specs).

The current release of the TSDIS Toolkit will be supported on three platforms, SGI, SUN, and HP. A future release will be provided for the DEC Alpha. This release includes the following functionality: routines to open and close HDF files (TKopen, TKclose); TKseek, which will position the file point to a specified place in the data product; TKreadScan, TKwriteScan, TKreadGrid, and TKwriteGrid for accessing both satellite and GV data; TKreportWarning and TKreport Error for sending messages to an error file; TKreadlsm, to read the Land Sea mask given a latitude and longitude; TKreadTopo, to read the digital elevation map (topographic database). Metadata access functions are provided and a TKendOfFile function will indicate when an end-of-file condition is encountered. TKreadHeader and TKwriteHeader provide access to PR ray headers and clutter flags.

1.1 TOOLKIT HINTS

There are a couple of important points that should be noted when installing and using the TSDIS toolkit. In the makefile provided with the release of the toolkit the paths that point to the various libraries, such as HDF, will have to be changed by the user to point to the HDF distribution

directories on your computer system. Follow the installation instructions included in the makefile for details of how to do this.

To be compatible across all supported platforms, Sun, HP, and SGI, all FORTRAN files must have file extensions ending in 'F' (upper case F) rather than 'f' (lower case f).

When a file is opened as TK_NEW_FILE, the metadata will be initialized for that product using the default values. These values can be changed individually by the developer. Changes to individual metadata elements can be performed using the metadata access routines discussed below.

2. TOOLKIT CATEGORIES

There are five basic categories of toolkit routines: Input/Output, Error Handling, Geolocation, Constants and Conversion Factors, and Mathematical and Statistical Routines. A detailed description of each routine can be found in Section 3, "TSDIS Toolkit Routine Specification". Section 3 contains two sections, 3.1 contains routine descriptions for C programmers, and section 3.2 contains routine descriptions for FORTRAN programmers.

Brief descriptions of the routines follow.

2.1 INPUT/OUTPUT

The Input and Output routines are designed to make it easy for the Algorithm Developer to access TRMM data. The routines are listed below, and fall into several classes: File Acces, Data Access (Scan), Data Access (Grid), Data Access (Level 1 GV), and Metadata Access.

File Access: TKopen, TKseek, TKclose, TKreadlsm, TKreadTopo, TKendOfFile

TKopen opens a file for reading or writing. Tkclose closes a file. TKseek points the file pointer to an specified scan in the file. TKreadlsm reads the land-sea mask database, returning the geography for a give coordinate point. TKreadTopo reads the ETOPO5 database, returning elevation for a given coordinate point. TKendOfFile signals when an end of file has been reached.

Data Access: (Scan) TKreadScan, TKwriteScan

TKreadScan reads a single scan from an open file containing scan based satellite. TKwriteScan writes a single scan to an open file containing scan based satellite.

Data Access: (Grid) TKreadGrid, TKwriteGrid

These routines read and write data for level 3 grid based satellite data products, and level 2 and 3 GV products.

Data Access: (L1 GV) TKgetNvos, TKgetNsensor, TKgetNparam, TKgetNcell, TKgetNray, TKgetNsweep, TKsetL1GVtemplate, TKreadL1GV, TKwriteL1GV

These routines access L1 GV data products. The TKgetNxxx routines provide information about the granule; TKsetL1GVtemplate creates a template for an output data product; TKreadL1GV and TKwriteL1GV read and write the L1 GV data.

Metadata Access: TKreadMetadataChar, TKwriteMetadataChar, TKreadMetadataFloat, TKwriteMetadataFloat,TKreadMetadataInt, TKwriteMetadataInt

There is a separate metadata routine for Character, Floating Point and Integer data types. The TKreadMetadataTYPE routines read a single metadata element into a typed variable. The TKwriteMetadataTYPE routines write a single metadata element to a file. Since the metadata is stored internally as characters, these routines translate from or to the appropriate type.

The metadata routines partially implemented: not all metadata elements are supported.

Header Access: TKreadHeader, TKwriteHeader

These routines read and write the ray header for PR L1B21 and L1C21 data products, and read and write clutter flags for L2A25 PR data products.

2.2 ERROR PROCESSING

There are two error processing routines that will be of interest to the Algorithm Developers. The purpose of these routines is to handle errors in a simple and consistent manner. Furthermore, when algorithms are running in the production environment or the ITE at TSDIS, these routines assist the TSDIS staff in diagnosing any problems.

TKreportError

This routine is used to process fatal errors. Calling this routine will close all open files, send an error message to a central log file, and terminate processing.

TKreportWarning

This routine is used for warnings and informational messages. Calling this routine will send an error message to a central log file and return control to the calling routine.

The error routines look for the error 'message file' in the directory $TSDISTK/include, where TSDISTK is an environment variable used by several toolkit routines. Details can be found in the toolkit installation instructions.

2.3 GEOLOCATION

The details of the geolocation routines are contained in the Level 1 Software Design Specification, and will be released in February of 1996.

2.4 CONSTANTS AND CONVERSION FACTORS

The constants and conversion factors consist of physical constants such as earth radius, factors for converting between degrees and radians, and time conversion routines reused from the ECS PGS toolkit.

An initial set of constants and conversion factors was part of release 2 of the TSDIS toolkit.

2.5 MATH AND STATISTICAL ROUTINES

The math and statistical routines will consist of IMSL. This library will be available in the TSDIS environment. Algorithms that need to call math or statistics routines should use the IMSL routines or supply their own code.

3. TSDIS TOOLKIT ROUTINE SPECIFICATION

This section includes a detailed description of each routine currently being developed for the TSDIS Toolkit. Each routine description briefly explains the purpose of the routine, the calling sequence, and the input and output parameters. A more detailed description of some parts of the routine is followed by a simple example of how the routine is used in working code. Each description ends by listing any prerequisites for using the routine, e.g., opening a file is a prerequisite to reading a scan.

The documentation of each routine contains the following:

NAME:	The name of the routine.
DESCRIPTION:	A brief description of what the routine does.
USAGE:	Example of the declaration and calling sequence of the routine.
INPUTS:	List and description of input parameters.
OUTPUTS:	List and description of output parameters.
DETAILS:	Relevant details of how the routine works and where to find more information.
EXAMPLES:	Short examples of how to use the routine in real code.
RETURN VALUES:	A brief list of common return values.
PREREQUISITES:	An explanation of what other routines need to be used in conjunction with the current routine.

The documentation of the individual Toolkit routines is in two parts. First, all the routines are listed in alphabetical order and are discussed with explanations of how to call them from C programs. This is the C version. The next part lists the same routines, in alphabetical order, with explanations of how to call them from FORTRAN programs. This is the FORTRAN version.
These sections are followed by the Parameter Dictionary, which contains descriptions of each of the parameters used in the toolkit routines. The Parameter Dictionary applies equally well to the C and FORTRAN toolkit routines.

3.1 ROUTINE SPECIFICATION, C VERSION

NAME: TKclose()

DESCRIPTION: This routine closes a data product file.

USAGE:
```
#include "IO.h"
int TKclose(IO_HANDLE *granuleHandle);
```

INPUTS: granuleHandle -
A structure containing information about the file which is to be closed. granuleHandle is returned by TKopen().

OUTPUTS: None.

DETAILS: Detailed descriptions of the input and output parameters and return codes can be found in the Appendix "Parameter Dictionary".

EXAMPLES:
```
#include "IO.h"
/* Include the I/O header file for TMI */
#include "IO_TMI.h"

/* Declare Variables */
IO_HANDLE      granuleHandle1B11;
int        status;

status = TKclose(&granuleHandle1B11);

/* Check the Error Status */
if (status != TK_SUCCESS)
/* handle error processing here */
```

RETURN VALUES:
```
TK_SUCCESS      - Routine completed successfully.
TK_FAIL         - Access routine failed.
```

PREREQUISITES: Before closing the file by TKclose(), the file should have been opened by TKopen().

NAME: TKopen()

DESCRIPTION: This routine opens a data product file prior to reading or writing product data or metadata.

USAGE:

```
#include "IO.h"

int TKopen(char *granuleID, int dataType,
        char filemode, IO_HANDLE *granuleHandle);
```

INPUTS:

granuleID -
A character string containing the name of the file to be opened. A maximum length of 255 characters is allowed.

dataType -
An unique identifier that specifies the type of data product being read, e.g., TK_L1B_11 for the 1B-11 algorithm product. A complete list of dataTypes can be found in the Parameter Dictionary.

filemode -
Access mode, TK_READ_ONLY and TK_NEW_FILE. TK_READ_ONLY opens the file in read only mode, without changing the metadata; TK_NEW_FILE opens the file in write only mode but initializes the metadata with default values. The metadata initialization is not currently implemented.

OUTPUTS:

granuleHandle -
A structure passed to subsequent I/O routines. This structure is manipulated internally by TKopen() and other toolkit routines.

DETAILS:

All files opened by TKopen() must be closed by TKclose(). TKopen() must be called prior to reading or writing to a file with a toolkit routine. When a file is opened, the file pointer is positioned at the beginning of the orbit, not the beginning of the granule. The file pointer can be repositioned using Tkseek().

Opening a file with filemode=TK_NEW_FILE will initialize the metadata for that file with default values and allow write-only access to the file. These values can be changed using one of the metadata access routines.

A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

EXAMPLES:

```
#include "IO.h"
/* Include the I/O header file for TMI */
#include "IO_TMI.h"

/* Declare Variables */

IO_HANDLE       granuleHandle1B11;
char            granuleID[TK_FNAME_LEN];
int             dataType;
char            filemode;
int             status;

/* Open the file for reading */
strcpy(granuleID, "L1B_11_input.dat");
dataType = TK_L1B_11;
filemode = TK_READ_ONLY;
status = TKopen(granuleID, dataType, filemode,
        &granuleHandle1B11);

/* Check the Error Status */
if (status != TK_SUCCESS)
        /* handle error processing here */
```

RETURN VALUES:

TK_SUCCESS	- Routine completed successfully.
TK_FAIL	- Open routine failed.

PREREQUISITES: None.

NAME: TKreadGrid()

DESCRIPTION: This routine reads gridded data from Level 3 TRMM satellite and L2 and L3 GV products.

USAGE:
```
#include "IO.h"
int TKreadGrid(IO_HANDLE *granuleHandle,
void *planetGrid);
```

INPUTS: granuleHandle -
A structure containing information about the file from which data can be obtained. granuleHandle is returned by TKopen().

OUTPUTS: planetGrid -
A structure containing the complete grid data which is obtained from the input data product. This structure must be declared to correspond to the data product being read.

DETAILS: A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

EXAMPLES:
```
#include "IO.h"
/* Include the I/O header file for TMI */
#include "IO_TMI.h"

/* Declare Variables */

L3A_11_PLANETGRID     l3A11Grid;
IO_HANDLE             granuleHandle3A11;
char                  granuleID[TK_FNAME_LEN];
int                   dataType;
char                  filemode;
int                   status;

/* Access the file for reading */
strcpy(granuleID, "L3A_11_input.dat");
dataType = TK_L3A_11;
filemode = TK_READ_ONLY;
status = TKopen(granuleID, dataType,
        filemode, &granuleHandle3A11);
```

```
/* Check the Error Status */
if (status != TK_SUCCESS)
        /* handle error processing here */

status = TKreadGrid(&granuleHandle3A11, &l3A11Grid);

/* Check the Error Status */
if (status != TK_SUCCESS)
        /* handle error processing here */
```

RETURN VALUES:

TK_SUCCESS	- Routine completed successfully.
TK_FAIL	- Routine failed.

PREREQUISITES: Before calling TKreadGrid(), a file must be opened for reading by calling TKopen(). When the file is no longer needed, it should be closed by calling TKclose().

NAME: TKreadlsm()

DESCRIPTION: Reads the land-sea database and returns a code that specifies if a given lat-lon point is land, ocean, coast, ice, or coast next to ice.

USAGE:
```
#include "landsea.h"
int TKreadlsm(float lat, float lon);
```

INPUTS:

lat -
Latitude coordinate, in degrees, of the point to be tested. Latitude range is from -90 degrees to +90 degrees.

lon -
Longitude of the coordinate, in degrees, of the specified point. Longitude range is from 0 to 359 degrees. Negative longitudes greater between -180 and 0 degrees are accepted and translated to positive values.

OUTPUTS: None.

DETAILS: The data file must be located in the directory '$TSDISTK/data'. The data filename is 'dbglobe93.grd'.

EXAMPLES:
```
/* Include the Land-sea header file */
 #include "landsea.h"

/* Declare Variables */
float   lat, lon;
int     lstype;

/* Read the land sea data file for a give latitude and
*longitude
*/
lat = 23.0;
lon = 5.0;
lstype = TKreadlsm(lat, lon);

 if (lstype == TK_LAND)
        /* do processing when over land */
 else if (lstype == TK_OCEAN)
/* do processing when over ocean */
else if (lstype == TK_COAST)
        /* do processing when along coast */
```

RETURN VALUES: The values returned are TK_LAND (land), TK_ICE (ice), TK_OCEAN (ocean), TK_COAST (coast), and TK_CICE (coast next to ice).

PREREQUISITES: None.

NAME: TKreadMetadataChar()

DESCRIPTION: This routine reads individual character metadata items from an HDF data product.

USAGE:
```
#include "IO.h"

int TKreadMetadataChar(IO_HANDLE *granuleHandle,
        char *parameter,
        char *value);
```

INPUTS: granuleHandle -
A structure containing information about the data product from which the metadata is read. granuleHandle is returned by TKopen().

parameter -
A string containing the name of the metadata element to be read from the file.

OUTPUTS: value -
The character value corresponding to the metadata element specified by 'parameter'.

DETAILS: All metadata is stored internally in character format.
The metadata access routines translate the character data to the appropriate format as needed. A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

EXAMPLES:
```
#include "IO.h"
/* Include the I/O header file for TMI */
#include "IO_TMI.h"

/* Declare Variables */
IO_HANDLE       granuleHandle1B11;
char            granuleID[TK_FNAME_LEN];
int             dataType;
 char           filemode;
int             status;
char            algID[TK_ALGID_LEN];
```

```
/* Access the file for reading */
strcpy(granuleID, "L1B_11_input.dat");
dataType = TK_L1B_11;
filemode = TK_READ_ONLY;
status = TKopen(granuleID, dataType, filemode,
        &granuleHandle1B11);

/* Check the Error Status */
if (status != TK_SUCCESS)
        /* handle error processing here */

status = TKreadMetadataChar(&granuleHandle1B11,
        TK_ALGORITHM_ID,
                algID);

/* Check the Error Status */
if (status != TK_SUCCESS)
        /* handle error processing here */
```

RETURN VALUES: TK_SUCCESS - Routine completed successfully.
TK_FAIL - Routine failed.

PREREQUISITES: Before calling TKreadMetadataChar(), a data product must be opened for reading by calling TKopen(). When the data product is no longer needed, it should be closed by calling TKclose().

NAME: TKreadMetadataFloat()

DESCRIPTION: This routine reads floating point metadata items from an HDF data product.

USAGE:
```
#include "IO.h"
int TKreadMetadataFloat(IO_HANDLE *granuleHandle,
char *parameter,
float *value);
```

INPUTS: granuleHandle -
A structure containing information about the data product from which the metadata is retrieved.granuleHandle is returned by TKopen().

parameter -
A string containing the name of the metadata element be read from the data product.

OUTPUTS: value -
A floating point value corresponding to the metadata element specified by 'parameter'.

DETAILS: All metadata is stored internally in character format.
The metadata access routines translate the character data to the appropriate format as needed. Detailed descriptions of the input and output parameters and return codes can be found in the Appendix "Parameter Dictionary".

EXAMPLES:
```
#include "IO.h"
/* Include the I/O header file for TMI */
#include "IO_TMI.h"

/* Declare Variables */
IO_HANDLE       granuleHandle1B11;
char            granuleID[TK_FNAME_LEN];
int             dataType;
char            filemode
int             status;
float           percentBadMissPix;
```

```
/* Access the file for reading */
strcpy(granuleID, "L1B_11_input.dat");
dataType = TK_L1B_11;
filemode = TK_READ_ONLY;
status = TKopen(granuleID, dataType, filemode,
        &granuleHandle1B11);

/* Check the Error Status */
if (status != TK_SUCCESS)
        /* handle error processing here */

status = TKreadMetadataFloat(&granuleHandle1B11,
        TK_PERCENT_BAD_MISS_PIXEL,
        &percentBadMissPix);

/* Check the Error Status */
if (status != TK_SUCCESS)
        /* handle error processing here */
```

RETURN VALUES:

TK_SUCCESS	- Routine completed successfully.
TK_FAIL	- Routine failed.

PREREQUISITES: Before calling TKreadMetadataFloat(), a data product must be opened for reading by calling TKopen(). When the data product is no longer needed, it should to be closed by calling TKclose().

NAME: TKreadMetadataInt()

DESCRIPTION: This routine reads individual integer metadata items from an HDF data product.

USAGE:
```
#include "IO.h"
int TKreadMetadataInt(IO_HANDLE *granuleHandle,
char *parameter,
void *value);
```

INPUTS: granuleHandle -
A structure containing information about the file from which the metadata is to be read. granuleHandle is returned by TKopen().

parameter -
A string containing the name of the metadata element read from the data product.

OUTPUTS: value -
A integer value corresponding to the metadata element specified by 'parameter'.

DETAILS: All metadata is stored internally in character format. The metadata access routines translate the character data to the appropriate format as needed. Detailed descriptions of the input and output parameters and return codes can be found in the Appendix "Parameter Dictionary".

EXAMPLES:
```
#include "IO.h"
/* Include the I/O header file for TMI */
#include "IO_TMI.h"

/* Declare Variables */
IO_HANDLE       granuleHandle1B11;
char            granuleID[TK_FNAME_LEN];
int             dataType;
char            filemode;
int             orbitNum;
int             status;
```

```
/* Access the file for reading */
strcpy(granuleID, "L1B_11_input.dat");
dataType = TK_L1B_11;
filemode = TK_READ_ONLY;
status = TKopen(granuleID, dataType, filemode,
        &granuleHandle1B11);

/* Check the Error Status */
if (status != TK_SUCCESS)
        /* handle error processing here */

status = TKreadMetadataInt(&granuleHandle1B11,
        TK_ORBIT_NUMBER,
        &orbitNum);

/* Check the Error Status */
if (status != TK_SUCCESS)
        /* handle error processing here */
```

RETURN VALUES:

TK_SUCCESS	- Routine completed successfully.
TK_FAIL	- Routine failed.

PREREQUISITES: Before calling TKreadMetadataInt(), a data product must be opened for reading by calling TKopen(). When the data product is no longer needed, it should be closed by calling TKclose().

NAME: TKreadScan()

DESCRIPTION: This routine reads scan based satellite product data and stores them into the product data structure.

USAGE:
```
#include "IO.h"
int TKreadScan(IO_HANDLE *granuleHandle,
void *swathData);
```

INPUTS: granuleHandle -
A structure containing information about the file from which data will be read. granuleHandle is returned by TKopen().

OUTPUTS: swathData -
A structure containing the complete scanline data which is obtained from the input data product. This structure must be declared to correspond to the data file being read.

DETAILS: TKopen positions the file pointer at the beginning of the orbit, not the granule. Thus, the first call to readScan returns the first scan in the orbit. The scan number is updated on each call so consecutive calls return consecutive scans. TKseek() may be used to move forward or backward by a certain number of scans.

Detailed descriptions of the input and output parameters and return codes can be found in the Appendix "Parameter Dictionary".

EXAMPLES:
```
#include "IO.h"
/* Include the I/O header file for TMI */
#include "IO_TMI.h"

/* Declare Variables */
L1B_11_SWATHDATA      l1B11Data;
IO_HANDLE             granuleHandle1B11;
char                  granuleID[TK_FNAME_LEN];
int                   dataType;
char                  filemode;
int                   status;
```

```
/* open the file */
strcpy(granuleID, "L1B_11_input.dat");
dataType = TK_L1B_11;
filemode = TK_READ_ONLY;
status = TKopen(granuleID, dataType, filemode,
        &granuleHandle1B11);

/* Check the Error Status */
if (status != TK_SUCCESS)
        /* handle error processing here */

status = TKreadScan(&granuleHandle1B11,
&l1B11Data);

/* Check the Error Status */
if (status != TK_SUCCESS)
        /* handle error processing here */
```

RETURN VALUES:

TK_SUCCESS	- Routine completed successfully.
TK_FAIL	- Routine failed.

PREREQUISITES: Before calling TKreadScan(), a file must be opened for reading by calling TKopen(). When the file is no longer needed, it should be closed by calling TKclose().

NAME: TKreadTopo()

DESCRIPTION: This routine reads the ETOPO5 database, returning an elevation in meters for a given coordinate point.

USAGE:
```
#include "etopo5.h"
short int TKreadTopo(float lat, float lon)
```

INPUTS: lat -
Latitude coordinate, in degrees, of the point for which elevation is desired. Latitude runs from -90 degrees to +90 degrees.

lon -
Longitude of the coordinate, in degrees, of the point for which elevation is desired. Longitude runs from 0 to 359 degrees. Negative longitudes between -180 and 0 degrees are also accepted.

OUTPUTS: None.

DETAILS: The ETOPO5 database is gridded in 1/12 degree bins. The data file must be located in '$TSDISTK/data/etop05.dat'.

EXAMPLES:
```
/* Include the topo header file */
#include "etopo5.h"

/* Declare Variables */
float       lat, lon;
short int   elevation;

/* Read the ETOPO5 data file for a give latitude and
* longitude
*/
lat = 50.0;
lon = 17.0;
elevation = TKreadTopo(lat, lon);

/* Do processing */
```

RETURN VALUES: Returns the elevation of the coordinate point in meters. When over the ocean, the elevation may be negative.

PREREQUISITES: None.

NAME: TKreportError()

DESCRIPTION: This routine records an error message based on the error number provided by the user, or a status returned by a routine. It then closes files opened by TKopen, and stops processing.

USAGE:
```
#include "TS_XX_YY.h"
void TKreportError(int errorNumber)
```

INPUTS: errorNumber -
A parameter that is used to index a particular error message. Each error number corresponds to a specific error message.

OUTPUTS: None.

DETAILS: The complete list of error codes can be found in the Parameter Dictionary.

This routine records the error message in a log file, prints it to the console, closes all files opened by TKopen, and terminates the process.

When TKreportError is called by the user, it expands as a macro to a different routine that includes '__LINE__' and '__FILE__' after the errorNumber. These two additional arguments capture the line number and file name of the call to TKreportError and are printed to the console and the log file along with the error message.

The error messages are in files named like "TS_TK_15.h", where TS stands for TSDIS, TK stands for Toolkit, and 15 means that all the error numbers start at 15000. The appropriate include file must be #included in the source code you write for the error routines to workcorrectly.

The routine then looks for the error messages in the file "TS_15", which must be located in the directory $TSDISTK/include.

EXAMPLES: This example shows the preprocessing steps, the variable declaration, and calls to TKreportError with two different conditions. The parameter in the call to TKreportError is the error number.

```
/* Include files */
 #include "IO.h"          /* for TK_SUCCESS */
 #include "TS_XX_YY.h"
```

```
/* Declare Variables */
 int            status;
 int            nr;

/*
 * Call a processing routine that returns an error
 * status.
 */
 status = process_data();

/*
 * Check if the status is a success, if not a success
 * then report the fatal error.  E_TK_INVALID_DATA is
 * a TSU defined message that has been registered with
  * TSDIS.
 */
if (status != TK_SUCCESS)
 TKreportError(E_TK_INVALID_DATA);

...

/*
 * Call a user defined processing routine that returns
 * the number of records read.
 */
nr = read_records();

/*
 * If the number of records read is less than the
 * expected number of records, NUM_REC, then report
 * the error E_TK_READ_FAIL.  The error message,
 * E_TK_READ_FAIL, are TSU defined messages.  Any
 * messages that are reported with TKreportError
 * should be registered with TSDIS.
 */

if (nr < NUM_REC)
        TKreportError(E_TK_READ_FAIL);
```

RETURN VALUES: None.

PREREQUISITES: None.

NAME: TKreportWarning()

DESCRIPTION: This routine reports a warning message based on a warning number provided by the user, or a status returned by a routine. Control is returned to the calling program.

USAGE:
```
#include "TS_XX_YY.h"
void TKreportWarning(int warnNumber)
```

INPUTS: warnNumber -
A parameter that is used to identify a particularwarning message. Each warning number corresponds to a specific message.

OUTPUTS: None.

DETAILS: The complete list of warning codes can be found in the Parameter Dictionary.

The routine records the corresponding warning message in a log file, prints the message to the console, and returns control to the calling program. Compare with TKreportError().

When TKreportWarning is called by the user, it expands as a macro to a different call that includes '__LINE__' and '__FILE__' after the warnNumber. These two additional arguments capture the line number and file name of the call to TKreportWarning and are printed to the operator and the log file along with the error message.

The error messages are in files named like "TS_TK_15.h", where TS stands for TSDIS, TK stands for Toolkit, and 15 means that all the error numbers start at 15000. The appropriate include file must be #included in the source code you write for the error routines to workcorrectly.

The routine then looks for the error messages in the file "TS_15", which must be located in the directory $TSDISTK/include.

EXAMPLES: This example shows the preprocessing steps, the variable declaration, and a call to TKreportWarning with two different conditions. The parameter in the call to TKreportWarning is the warning number.

```
/* Include files */
#include "TS_XX_YY.h"
#include "IO.h"
```

```
/* Declare Variables */
int             status;
int             nr;

/*
 * Call a processing routine that returns an error
 * status.
 */
status = process_data();

/* Check if the status is a success, if not a success
 * then report the warning error.  W_TK_INVALID_DATA
 * is a TSU defined message that has been registerd
 * with TSDIS.
 */
if (status != TK_SUCCESS)
        TKreportWarning(W_TK_INVALID_DATA);

...

/* Call a user defined processing routine that returns
 * the number of records read.
 */
nr = read_records();

/* if the number of records read is less than the
 * expected number of records, NUM_REC, then report the
 * warning, W_READ_FAIL.
 */

if (nr < NUM_REC)
        TKreportWarning(W_TK_READ_FAIL);
```

RETURN VALUES: None.

PREREQUISITES: None.

NAME: TKseek()

DESCRIPTION: This routine moves the file pointer to a specified position within the file; this enables reading of an specified scan line.

USAGE:
```
#include "IO.h"
int TKseek(IO_HANDLE *granuleHandle,
        int offset,
        int type);
```

INPUTS: granuleHandle -
A structure containing information about an open file. granuleHandle is returned by TKopen().

offset -
The specification of this parameter depends on the value of the 'type' parameter.

If type = TK_REL_SCAN_OFF, offset is an integer specifying the number of scan lines to move, relative to the current scan. If offset=5, this would move the file pointer forward by 5 scans. If offset=-1 the file pointer would move back by 1 scan.

If type = TK_ABS_SCAN_OFF, offset is an integer specifying the scan line relative to the beginning of the file.

type -
The type parameter will be one of two values. If TK_REL_SCAN_OFF is used as the type parameter, then the seek is relative to the current scan at the file pointer. If TK_ABS_SCAN_OFF is used, then the seek is from the beginning of the data product.

OUTPUTS: None

DETAILS: Detailed descriptions of the input and output parameters and return codes can be found in the Appendix "Parameter Dictionary".

EXAMPLES:
```
#include "IO.h"

/* Declare Variables */
```

```
IO_HANDLE       granuleHandle1B11;
char            granuleID[TK_FNAME_LEN];
int             dataType;
char            filemode;
int             offset;
int             status;

/* access the file for reading */
strcpy(granuleID, "L1B_11_input.dat");
dataType = TK_L1B_11;
filemode = TK_READ_ONLY;
status = TKopen(granuleID, dataType, filemode,
        &granuleHandle1B11);

/* Check the Error Status */
if (status != TK_SUCCESS)
        /* handle error processing here */

/* Move the file pointer to the beginning of the
 * Granule
 */
offset = 5;
status = TKseek(&granuleHandle1B11, offset, TK_ABS_SCAN_OFF);

/* Check the Error Status */
if (status != TK_SUCCESS)
        /* handle error processing here */
```

RETURN VALUES:

TK_SUCCESS	- Offset was successful
W_TK_BADOFSET	- Requested offset was out of range

PREREQUISITES: Before using TKseek(), the file must have been opened by TKopen().

NAME: TKwriteGrid()

DESCRIPTION: This routine writes gridded data to Level 3 TRMM satellite and Level 2 and Level 3 GV data products.

USAGE:
```
#include "IO.h"
int TKwriteGrid(IO_HANDLE *granuleHandle,
void *planetGrid);
```

INPUTS:

granuleHandle -
A structure containing information about the file from which data can be obtained. granuleHandle is returned by TKopen().

planetGrid -
A structure containing the complete grid data which is obtained from the input file. This structure must be declared to correspond to the data file being read.

OUTPUTS: None.

DETAILS: A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

EXAMPLES:
```
#include "IO.h"
/* Include the I/O header file for TMI */
#include "IO_TMI.h"

/* Declare Variables */
L3A_11_PLANETGRID      l3A11Grid;
IO_HANDLE              granuleHandle3A11;
char                   granuleID[TK_FNAME_LEN];
int                    dataType;
char                   filemode;
 int                   status;

/* Access the file for writing */
strcpy(granuleID, "L3A_11_output.dat");
dataType = TK_L3A_11;
filemode = TK_NEW_FILE;
status = TKopen(granuleID, dataType, filemode,
        &granuleHandle3A11);
```

```
/* Check the Error Status */
if (status != TK_SUCCESS)
        /* handle error processing here */

status = TKwriteGrid(&granuleHandle3A11, &l3A11Grid);

/* Check the Error Status */
if (status != TK_SUCCESS)
/* handle error processing here */
```

RETURN VALUES:

TK_SUCCESS	- Routine completed successfully.
TK_FAIL	- Routine failed.

PREREQUISITES: Before calling TKwriteGrid(), a file must be opened for reading by calling TKopen(). When the file is no longer needed, it should be closed by calling TKclose().

NAME: TKwriteMetadataChar()

DESCRIPTION: This routine writes individual character metadata items to an HDF data product.

USAGE:

```
#include "IO.h"

int TKwriteMetadataChar(IO_HANDLE *granuleHandle,
        char *parameter,
        char *value);
```

INPUTS:

granuleHandle -
A structure containing information about the file to which metadata information can be written. granuleHandle is returned by TKopen().

parameter -
A string containing the name of the metadata element to be read from the file.

value - The character value corresponding to the metadata element specified by parameter.

OUTPUTS: None.

DETAILS: All metadata is stored internally in character format. The metadata access routines translate the character data to the appropriate format as needed. Detailed descriptions of the input and output parameters and return codes can be found in the Appendix "Parameter Dictionary".

EXAMPLES:

```
#include "IO.h"
/* Include the I/O header file for TMI */
#include "IO_TMI.h"

/* Declare Variables */
IO_HANDLE       granuleHandle1B11;
char            granuleID[TK_FNAME_LEN];
int             dataType;
char            filemode;
int             status;
char            algID[TK_ALGID_LEN];
```

```
/* Access the file for writing */
strcpy(granuleID, "L1B_11_output.dat");
dataType = TK_L1B_11;
filemode = TK_NEW_FILE;
status = TKopen(granuleID, dataType, filemode,
        &granuleHandle1B11);

/* Check the Error Status */
if (status != TK_SUCCESS)
        /* handle error processing here */

strcpy(algID, "TK_L1B_11");

status = TKwriteMetadataChar(&granuleHandle1B11,
        TK_ALGORITHM_ID,
        algID);

/* Check the Error Status */
if (status != TK_SUCCESS)
        /* handle error processing here */
```

RETURN VALUES:

TK_SUCCESS	- Routine completed successfully.
TK_FAIL	- Routine failed.

PREREQUISITES: Before calling TKwriteMetadataChar(), a data product has to be opened for writing by calling Tkopen(). When the data product is no longer needed, it should be closed by calling TKclose().

NAME: TKwriteMetadataFloat()

DESCRIPTION: This routine writes individual floating point metadata elements to an HDF data product.

USAGE:
```
#include "IO.h"
int TKwriteMetadataFloat(IO_HANDLE *granuleHandle,
        char *parameter,
        float *value);
```

INPUTS:

granuleHandle -
A structure containing information about the data product to which metadata information can be written. granuleHandle is returned by TKopen().

parameter -
A string containing the name of the metadata element to be retrieved from the data product.

value -
The character value corresponding to the metadata element specified by parameter.

OUTPUTS: None.

DETAILS: All metadata is stored internally in character format. The metadata access routines translate the character data to the appropriate format as needed. Detailed description of the input and output parameters and return codes can be found in the Appendix "Parameter Dictionary".

EXAMPLES:
```
#include "IO.h"
/* Include the I/O header file for TMI */
#include "IO_TMI.h"

/* Declare Variables */
IO_HANDLE       granuleHandle1B11;
char            granuleID[TK_FNAME_LEN];
int             dataType;
char            filemode;
int             status;
float           percentBadMissPix;
```

```
/* Access the file */
strcpy(granuleID, "L1B_11_output.dat");
dataType = TK_L1B_11;
filemode = TK_NEW_FILE;
status = TKopen(granuleID, dataType, filemode,
        &granuleHandle1B11);

/* Check the Error Status */
if (status != TK_SUCCESS)
        /* handle error processing here */

percentBadMissPix = 20.3;

status = TKwriteMetadataFloat(&granuleHandle1B11,
        TK_PERCENT_BAD_MISS_PIXEL,
        &percentBadMissPix);

/* Check the Error Status */
if (status != TK_SUCCESS)
/* handle error processing here */
...
```

RETURN VALUES:

TK_SUCCESS	- Routine completed successfully.
TK_FAIL	- Routine failed.

PREREQUISITES: Before calling writeMetadataFloat(), a data product must be opened for writing by calling TKopen(). When the data product is no longer needed, it should be closed by calling TKclose().

NAME: TKwriteMetadataInt()

DESCRIPTION: This routine writes individual integer metadata items to an HDF data product.

USAGE:
```
#include "IO.h"
int TKwriteMetadataInt(IO_HANDLE *granuleHandle,
char *parameter,
void *value);
```

INPUTS:

granuleHandle -
A structure containing information about the data product to which metadata information can be written. granuleHandle is returned by TKopen().

parameter -
A string which containing the name of the metadata element to be written to the data product.

value -
The character value corresponding to the metadata element specified by parameter.

OUTPUTS: None.

DETAILS: All metadata is stored internally in character format. The metadata access routines translate the character data to the appropriate format as needed. Detailed description of the input and output parameters and return codes can be found in the Appendix "Parameter Dictionary".

EXAMPLES:
```
#include "IO.h"

/* Declare Variables */
IO_HANDLE       granuleHandle2A12;
 char           granuleID[TK_FNAME_LEN];
int             dataType;
char            filemode;
int             status;
 int            orbitNum;
```

```
/* Access the file for writing */
strcpy(granuleID, "L2A_12_output.dat");
dataType = TK_L2A_12;
filemode = TK_NEW_FILE;
status = TKopen(granuleID, dataType, filemode,
        &granuleHandle2A12;

/* Check the Error Status */
if (status != TK_SUCCESS)
        /* handle error processing here */

orbitNum = 104;

status = TKwriteMetadataInt(&granuleHandle1B11,
        TK_ORBIT_NUMBER,
        &orbitNum);

/* Check the Error Status */
if (status != TK_SUCCESS)
        /* handle error processing here */
```

RETURN VALUES:

TK_SUCCESS	- Routine completed successfully.
TK_FAIL	- Routine failed.

PREREQUISITES: Before calling TKwriteMetadataInt(), a data product must be opened for writing by calling TKopen(). When the data product is no longer needed, it should be closed by calling TKclose().

NAME: TKwriteScan()

DESCRIPTION: This routine writes scan based satellite data to a data product.

USAGE:
```
#include "IO.h"
int TKwriteScan(IO_HANDLE *granuleHandle,
void *swathData)
```

INPUTS: GranuleHandle -
A structure containing information about the data product to which data can be written. granuleHandle is returned by TKopen().

swathData -
A structure containing the complete scanline data which is to be written to the output data product. This structure must be declared to correspond to the data file being written.

OUTPUTS: None.

DETAILS: Each call to TKwriteScan increments the scan line number by one, so the output data product is appended with each scan written.

Detailed description of the input and output parameters and return codes can be found in the Appendix "Parameter Dictionary".

EXAMPLES:
```
#include "IO.h"
/* Include the I/O header file for TMI */
#include "IO_TMI.h"

L1B_11_SWATHDATA      l2A12Data;
IO_HANDLE             granuleHandle1B11;
char                  granuleID[TK_FNAME_LEN];
int                   dataType;
char                  filemode;
int                   status;

/* Open the file for writing */
strcpy(granuleID,'L2A_12_output.dat');
dataType = TK_L2A_12;
filemode = TK_NEW_FILE;
status = TKopen(granuleID, dataType, filemode,
        &granuleHandle2A12);
```

```
/* Check the Error Status */
if (status != TK_SUCCESS)
        /* Perform error handling here */

status = TKwriteScan(&granuleHandle1B11,
        &l2A12Data);

/* Check the Error Status */
if (status != TK_SUCCESS)
        /* perform error handling here */
```

RETURN VALUES:

TK_SUCCESS	- Routine completed successfully.
TK_FAIL	- Routine failed.

PREREQUISITES: Before calling TKwriteScan(), a data product must be opened by calling TKopen(). When the data product is no longer needed, it should be closed by calling TKclose().

NAME: TKgetNvos()

DESCRIPTION: This routine returns the number of Volume Scans in the GV granule.

USAGE:
```
#include "IO.h"
#include "IO_GV.h"

int TKgetNvos(IO_HANDLE *granuleHandle)
```

INPUTS: granuleHandle -
A structure containing information about the data product to which data can be written. granuleHandle is returned by TKopen().

OUTPUTS: None.

DETAILS: A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

EXAMPLES:
```
/* IO.h contains the function prototypes for the
 * majority of the I/O routines, including the GV
 * read and write routines.
 */
#include "IO.h"

/* Include the GV structure definitions and macros */
#include "IO_GV.h"

L1B_1C_GV       l1BGVData;
IO_HANDLE       granuleHandle1BGV;
char            granuleID[TK_FNAME_LEN];
int             dataType;
char            filemode;
int             status;
int             nvos;

/* Open the file for reading */
strcpy(granuleID,"L1B_GV_input.dat");
dataType = TK_L1B_GV;
filemode = TK_READ_ONLY;
status = TKopen(granuleID, dataType, filemode,
        &granuleHandle1BGV);
```

```
/* Check the Error Status */
if (status != TK_SUCCESS)
/* Perform error handling here */

nvos = TKgetNvos(&granuleHandle1BGV);

if (nvos <= 0)
/* Perform error handling here */
```

RETURN VALUES: The number of volume scans is returned as an integer, or TK_FAIL if the routine failed.

PREREQUISITES: The file associated with granuleHandle must have been previously opened by TKopen().

NAME: TKgetNsensor()

DESCRIPTION: This routine returns the number of sensors in a GV VOS.

USAGE:
```
#include "IO.h"
#include "IO_GV.h""
int TKgetNsensor(IO_HANDLE *granuleHandle)
```

INPUTS: granuleHandle -
A structure containing information about the file to which data can be written. granuleHandle is returned by TKopen().

OUTPUTS: None.

DETAILS: A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

EXAMPLES:
```
/* IO.h contains the function prototypes for the
 * majority of the I/O routines, including the GV
 * read and write routines.
 */
#include "IO.h"

/* Include the GV structure definitions and macros */
#include "IO_GV.h"

L1B_1C_GV       l1BGVData;
IO_HANDLE       granuleHandle1BGV;
char            granuleID[TK_FNAME_LEN];
int             dataType;
char            filemode;
int             status;
int             nsensor;

/* Open the file for reading */
strcpy(granuleID,"L1B_GV_input.dat");
dataType = TK_L1B_GV;
filemode = TK_READ_ONLY;
status = TKopen(granuleID, dataType, filemode,
        &granuleHandle1BGV);
```

```
/* Check the Error Status */
if (status != TK_SUCCESS)
        /* Perform error handling here */

nsensor = TKgetNsensor(&granuleHandle1BGV);
```

RETURN VALUES: Returns number of sensors if successful, or TK_FAIL if the routine failed.

PREREQUISITES: The file corresponding to the granuleHandle must have been opened previously by TKopen().

NAME: TKgetNparam()

DESCRIPTION: This routine returns the number of parameters in a GV volume scan.

USAGE:
```
#include "IO.h"
#include "IO_GV.h"

int TKgetNparam(IO_HANDLE *granuleHandle,
        int VOSnumber);
```

INPUTS:

granuleHandle -
A structure containing information about the file to which data can be written. granuleHandle is returned by TKopen().

VOSnumber -
The volume scan number. This is a sequential number from 0 to the number of VOS in the granule.

OUTPUTS: None.

DETAILS: A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

EXAMPLES:
```
/* IO.h contains the function prototypes for the
 * majority of the I/O routines, including the GV
 * read and write routines.
 */
#include "IO.h"

/* Include the GV structure definitions and macros */
#include "IO_GV.h"

L1B_1C_GV       l1BGVData;
IO_HANDLE       granuleHandle1BGV;
char            granuleID[TK_FNAME_LEN];
int             dataType;
char            filemode;
int             status;
int             nparam;
int             nvos;
```

```
/* Open the file for reading */
strcpy(granuleID,'L1B_GV_input.dat');
dataType = TK_L1B_GV;
filemode = TK_READ_ONLY;
status = TKopen(granuleID, dataType, filemode,
&granuleHandle1BGV);

/* Check the Error Status */
if (status != TK_SUCCESS)
/* Perform error handling here */

nvos = 5;
nparam = TKgetNparam(&granuleHandle1BGV, nvos);
```

RETURN VALUES: Returns the number of parameters in the VOS if successful. Returns TK_FAIL if unsuccessful.

PREREQUISITES: The file corresponding to the granuleHandle must have been opened previously by TKopen().

NAME: TKgetNcell()

DESCRIPTION: This routine returns the number of cells for a given volume scan number and parameter number.

USAGE:
```
#include "IO.h"
#include "IO_GV.h"
int TKgetNcell(IO_HANDLE *granuleHandle, int
VOSnumber, int paramNum);
```

INPUTS:

granuleHandle -
A structure containing information about the file to which data can be written. granuleHandle is returned by TKopen().

VOSnumber -
The volume scan number. This is a sequential number from 0 to the number of VOSs in the granule.

paramNum -
The parameter number for the give volume scan.

OUTPUTS: None.

DETAILS: A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

EXAMPLES:
```
/* IO.h contains the function prototypes for the
 * majority of the I/O routines, including the GV
 * read and write routines.
 */
#include "IO.h"

/* Include the GV structure definitions and macros */
#include "IO_GV.h"

L1B_1C_GV       l1BGVData;
IO_HANDLE       granuleHandle1BGV;
char            granuleID[TK_FNAME_LEN];
int             dataType;
char            filemode;
int             status;
int             ncell;
int             vosNum;
int             paramNum;
```

```
/* Open the file for reading */
strcpy(granuleID,"L1B_GV_input.dat");
dataType = TK_L1B_GV;
filemode = TK_READ_ONLY;
status = TKopen(granuleID, dataType, filemode,
        &granuleHandle1BGV);

/* Check the Error Status */
if (status != TK_SUCCESS)
        /* Perform error handling here */

vosNum = 2;
paramNum = 2;
ncell = TKgetNcell(&granuleHandle1BGV, vosNum, paramNum );
```

RETURN VALUES: Returns the number of Cells in the VOS TK_FAIL - Routine Failed.

PREREQUISITES: File corresponding to granuleHandle must have been opened previously by calling TKopen().

NAME: TKgetNray()

DESCRIPTION: This routine returns the number of rays in a specified VOS.

USAGE:
```
#include "IO.h"
#include "IO_GV.h"
int TKgetNray(IO_HANDLE *granuleHandle, int VOSnum);
```

INPUTS: granuleHandle -
A structure containing information about the file to which data can be written. granuleHandle is returned by TKopen().

VOSnum -
The volume scan number. This is a sequential number form 0 to the number of VOSs in the granule.

OUTPUTS: None.

DETAILS: A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

EXAMPLES:
```
/* IO.h contains the function prototypes for the
 * majority of the I/O routines, including the GV
 * read and write routines.
 */
#include "IO.h"

/* Include the GV structure definitions and macros */
#include "IO_GV.h"

L1B_1C_GV       l1BGVData;
IO_HANDLE       granuleHandle1BGV;
char            granuleID[TK_FNAME_LEN];
int             dataType;
char            filemode;
int             status;
int             nray;
int             vosNum;

/* Open the file for reading */
strcpy(granuleID,"L1B_GV_input.dat");
dataType = TK_L1B_GV;
filemode = TK_READ_ONLY;
```

```
status = TKopen(granuleID, dataType, filemode,
        &granuleHandle1BGV);
/* Check the Error Status */
if (status != TK_SUCCESS)
/* Perform error handling here */

vosNum = 5;
nray = TKgetNray(&granuleHandle1BGV, vosNum );
```

RETURN VALUES: Returns the number of rays in the VOS, or TK_FAIL if unsuccessful.

PREREQUISITES: File corresponding to granuleHandle must have been opened previously by calling TKopen().

NAME: TKgetNsweep()

DESCRIPTION: This routine returns the number of sweeps for a given Volume Scan.

USAGE:
```
#include "IO.h"
#include "IO_GV.h"
int TKgetNsweep(IO_HANDLE *granuleHandle, int VOSnum);
```

INPUTS: granuleHandle -
A structure containing information about the file to which data can be written. granuleHandle is returned by TKopen().

VOSnum -
The volume scan number. This is a sequential number from 0 to the number of VOSs in the granule.

OUTPUTS: None.

DETAILS: A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

EXAMPLES:
```
/* IO.h contains the function prototypes for the
 * majority of the I/O routines, including the GV
 * read and write routines.
 */
#include "IO.h"

/* Include the GV structure definitions and macros */
#include "IO_GV.h"

L1B_1C_GV       l1BGVData;
IO_HANDLE       granuleHandle1BGV;
char            granuleID[TK_FNAME_LEN];
int             dataType;
char            filemode;
int             status;
int             nsweep;
int             vosNum;
```

```
/* Open the file for reading */
strcpy(granuleID,"L1B_GV_input.dat");
dataType = TK_L1B_GV;
filemode = TK_READ_ONLY;
status = TKopen(granuleID, dataType, filemode,
&granuleHandle1BGV);

/* Check the Error Status */
if (status != TK_SUCCESS)
        /* Perform error handling here */

vosNum = 2;
nsweep = TKgetNsweep(&granuleHandle1BGV, vosNum );
```

RETURN VALUES: Returns the number of sweeps in the volume scan or TK_FAIL if unsuccessful

PREREQUISITES: The volume scan corresponding to the granuleHandle must have been previously opened by TKopen().

NAME: TKsetL1GVtemplate()

DESCRIPTION: This routine creates a template for a GV product which is initialized with dimensions of the data product being written. This routine is used only for writing GV data products.

USAGE:
```
#include "IO.h"
#include "IO_GV.h"
int TKsetL1GVtemplate( int nvos, int
nparm[MAX_VOS], int ncell[MAX_VOS][MAX_PARM], int
nray[MAX_VOS], int nsweep[MAX_VOS], char *fileName);
```

INPUTS:

nvos -	Number of volume scans to create
nparam -	Number of parameters for each volume scan
ncell -	Array specifying the number of cells for each combination of volume scan and parameter.
nray -	Number of rays for each volume scan
nsweep -	Array specifying the number of sweeps for each volume scan.
fileName -	Name of the HDF file in which L1 GV data will be stored

OUTPUTS: None.

DETAILS: This routine must be called prior to calling TKopen when writing a L1 GV data product. It creates a template that is used in dimensioning the HDF file that will be written. The first time the routine is called, it creates the template file. Subsequent calls append new templates to the existing file.

A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

EXAMPLES:
```
/* IO.h contains the function prototypes for the
 * majority of the I/O routines, including the GV
 * read and write routines.
 */
#include "IO.h"

/* Include the GV structure definitions and macros */ #include "IO_GV.h"

L1B_1C_GV  l1BGVData;
IO_HANDLE granuleHandle1BGV;
char     granuleID[TK_FNAME_LEN];
```

```
int     dataType;
char    filemode;
int     status;
int     nvos;
int     nparm[MAX_VOS];
int     ncell[MAX_VOS][MAX_PARM];
int     nray[MAX_VOS];
int     nsweep[MAX_VOS];

/* Create a L1 GV template.  First set the values of
 * the GV file, then call the routine to create the
 * template file.
 */
nvos =       10;
nparm =    {2,2,2,2,2,2,2,2,2,2};
ncell =    {230,230,230,230,230,230,230,230,230,230,
             230,230,230,230,230,230,230,230,230,230};
nray =     {375,375,375,375,375,375,375,375,375,375};
nsweep =   {14,14,14,14,14,14,14,14,14,14};
strcpy(granuleID,"L1B_GV_input.dat");
status = TKsetL1GVtemplate(nvos, nparm, ncell, nray,
        nsweep, granuleID);

/* Check the error status */
if (status != TK_SUCCESS )
        /* Perform error handling here */

/* Open the file for writing */
dataType = TK_L1B_GV;
filemode = TK_NEW_FILE;
status = TKopen(granuleID, dataType, filemode,
        &granuleHandle1BGV);

/* Check the Error Status */
if (status != TK_SUCCESS)
/* Perform error handling here */
```

RETURN VALUES:

TK_SUCCESS	- If routine was successful
TK_FAIL	- If routine was unsuccessful

PREREQUISITES: This routine must be called prior to calling TKopen to open a GV granule for writing.

NAME: TKreadL1GV()

DESCRIPTION: This routine reads a L1 GV data product.

USAGE:
```
#include "IO.h"
#include "IO_GV.h"
int TKreadL1GV(IO_HANDLE *granuleHandle, void *sGV )
```

INPUTS: granuleHandle -
A structure containing information about the file which will be read. granuleHandle is returned by TKopen().

sGV -
A structure containing one volume scan data which is read from an HDF data product. This structure must be delcared to correspond to the data product being read.

OUTPUTS: None

DETAILS: A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

EXAMPLES:
```
/* IO.h contains the function prototypes for the
 * majority of the I/O routines, including the GV
 * read and write routines.
 */
#include "IO.h"

/* Include the GV structure definitions and macros */
#include "IO_GV.h"

L1B_1C_GV       l1BGVData;
IO_HANDLE       granuleHandle1BGV;
char            granuleID[TK_FNAME_LEN];
int             dataType;
char            filemode;
int             status;

/* Open the file for reading */
strcpy(granuleID,"L1B_GV_input.dat");
dataType = TK_L1B_GV;
filemode = TK_READ_ONLY;
status = TKopen(granuleID, dataType, filemode,
        &granuleHandle1BGV);
```

```
/* Check the Error Status */
if (status != TK_SUCCESS)
        /* Perform error handling here */

status = TKreadL1GV(&granuleHandle1BGV, &l1BGVData );

/* Check the Error status */
if (status != TK_SUCCESS)
        /* perform error handling here */
```

RETURN VALUES: TK_SUCCESS if the routine is successful. If an error occurs, TKreportError is called and processing is terminated.

PREREQUISITES: The file corresponding to granuleHandle must have been previously opened by TKopen.

NAME: TKwriteL1GV()

DESCRIPTION: This routine writes L1 GV granules to a product data file.

USAGE:
```
#include "IO.h"
int TKwriteL1GV(IO_HANDLE *granuleHandle, void *sGV)
```

INPUTS: granuleHandle -
A structure containing information about the file to which data can be written. granuleHandle is returned by TKopen().

OUTPUTS: sGV -
A structure containing one volume scan which is to be written to a GV HDF data product. This structure must be delcared to correspond to the data product being written.

DETAILS: A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

EXAMPLES:
```
/* IO.h contains the function prototypes for the
 * majority of the I/O routines, including the GV
 * read and write routines.
 */
#include "IO.h"

/* Include the GV structure definitions and macros */
#include "IO_GV.h"

L1B_1C_GV       l1BGVData;
IO_HANDLE       granuleHandle1BGV;
char            granuleID[TK_FNAME_LEN];
int             dataType;
char            filemode;
long int        status;
long int        nvos;
long int        nparm[MAX_VOS];
long int        ncell[MAX_VOS][MAX_PARM];
long int        nray[MAX_VOS];
long int        nsweep[MAX_VOS];

/* Create a L1 GV template.  First set the values of
 * the GV file, then call the routine to create the
 * template file.
 */
```

```
nvos =      10;
nparm =     {2,2,2,2,2,2,2,2,2,2};
ncell =     {230,230,230,230,230,230,230,230,230,230,
             230,230,230,230,230,230,230,230,230,230};
nray =      {375,375,375,375,375,375,375,375,375,375};
nsweep =    {14,14,14,14,14,14,14,14,14,14};
strcpy(granuleID,"L1B_GV_output.dat");
status = TKsetL1GVtemplate(nvos, nparm, ncell, nray,
          nsweep, granuleID);

/* Check the error status */
if (status != TK_SUCCESS )
     /* Perform error handling here */
/* Open the file for writing */
dataType = TK_L1B_GV;
filemode = TK_NEW_FILE;
status = TKopen(granuleID, dataType, filemode,
     &granuleHandle1BGV);

/* Check the Error Status */
if (status != TK_SUCCESS)
     /* Perform error handling here */

status = TKwriteL1GV( &granuleHandle1BGV,
&l1BGVData);
/* Check the Error status */
if (status != TK_SUCCESS)
     /* perform error handling here */
```

RETURN VALUES:

TK_SUCCESS	- Successful reading of data
TK_FAIL	- Routine failed

PREREQUISITES: To use TKwriteL1GV, the user must first call TKsetL1GVtemplate(), and then call TKopen(). There must be a separate call to TKsetL1GVtemplate() (and a corresponding call to TKopen) for each file being written with TKwriteL1GV().

NAME: TKendOfFile()

DESCRIPTION: This routine determines the number of records in an HDF file and returns TK_EOF when an end of file is reached.

USAGE:
```
#include "IO.h"
int TKendOfFile( IO_HANDLE *granuleHandle)
```

INPUTS: granuleHandle -
A structure containing information about the file to which data can be written. granuleHandle is returned by TKopen ().

OUTPUTS: None.

DETAILS: The first time this routine is called it determines the number of records in the HDF file. Each subsequent call to this routine decrements the number of records by one. When the number of records is zero, the routine returns TK_EOF, otherwise it returns TK_FAIL.

EXAMPLES:
```
/* IO.h contains the function prototypes for the
 * majority of the I/O routines, including the GV
 * read and write routines.
 */
#include "IO.h"

/* Include the I/O header file for TMI */
#include "IO_TMI.h"

L1B_11_SWATHDATA      l1B11Data;
IO_HANDLE             granuleHandle1B11;
char                  granuleID[TK_FNAME_LEN];
int                   dataType;
char                  filemode;
int                   status;

/* Open the file for reading */
strcpy(granuleID,"L1B_11_input.dat");
dataType = TK_L1B_11;
filemode = TK_READ_ONLY;
status = TKopen(granuleID, dataType, filemode,
        &granuleHandle1B11);
```

```
/* Check the Error Status */
if (status != TK_SUCCESS)
/* Perform error handling here */

while (TKendOfFile( &granuleHandle1B11 ) != TK_EOF) {
 status = TKreadScan(&granuleHandle1B11, &l1B11Data);

/* Check the Error Status */
 if (status != TK_SUCCESS)
/* perform error handling here */

 ...
}
```

RETURN VALUES: TK_EOF - Indicates an end of file has been reached. TK_FAIL - End of file conditon not encountered.

PREREQUISITES: The file corresponding to granuleHandle must have been previously opened for reading by TKopen().

NAME: TKreadHeader()

DESCRIPTION: This routine reads PR ray header data from a 1B21, 1C21 data product, or reads clutter flags from a 2A25 product data file.

USAGE:
```
#include "IO.h"
int TKreadHeader(IO_HANDLE *granuleHandle, void *sHeader);
```

INPUTS: granuleHandle -
A structure containing information about the file from which data can be obtained. granuleHandle is returned by TKopen().

OUTPUTS: sHeader -
A structure containing the PR calibration coefficients and ray header for 1B21 and 1C21 data products, or the clutter flags for the 2A25 data product.

DETAILS: A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

EXAMPLES:
```
#include "IO.h"
/* Include the I/O header file for PR */
#include "IO_PR.h"

/* Declare Variables */
L1B21_L1C21_HEADER    L1b21header;
IO_HANDLE             granuleHandle1B21;
char                  granuleID[TK_FNAME_LEN];
int                   dataType;
char                  filemode;
int       status;

/* Access the file for reading */
strcpy(granuleID, "L1B_21_input.dat");
dataType = TK_L1B_21;
filemode = TK_READ_ONLY;
status = TKopen(granuleID, dataType, filemode, &granuleHandle1B21);

/* Check the Error Status */
if (status != TK_SUCCESS)
/* handle error processing here */

status = TKreadHeader(&granuleHandle1b21, &L1b21header);
```

```
/* Check the Error Status */
if (status != TK_SUCCESS)
/* handle error processing here */
```

RETURN VALUES: TK_SUCCESS - Routine completed successfully.
W_TK_BADPIDRH - An invalid product ID was passed to TKreadHeader
W_TK_BADFILEMODRH - An invalid file mode was passed to TKreadHeader

PREREQUISITES: Before calling TKreadHeader(), a file must be opened for reading by calling TKopen(). When the file is no longer needed, it should be closed by calling TKclose().

NAME: TKwriteHeader()

DESCRIPTION: This routine writes PR ray header data to a 1B21, 1C21 data product, or writes clutter flags to a 2A25 product data file.

USAGE:
```
#include "IO.h"
int TKwriteHeader(IO_HANDLE *granuleHandle, void *sHeader);
```

INPUTS: granuleHandle -
A structure containing information about the file from which data can be obtained. granuleHandle is returned by TKopen().

OUTPUTS: sHeader -
A structure containing the PR calibration coefficients and ray header for 1B21 and 1C21 data products, or the clutter flags for the 2A25 data product.

DETAILS: A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

EXAMPLES:
```
#include "IO.h"
/* Include the I/O header file for PR */
#include "IO_PR.h"

/* Declare Variables */
L1B21_L1C21_HEADER    L1b21header;
IO_HANDLE             granuleHandle1B21;
char                  granuleID[TK_FNAME_LEN];
int                   dataType;
char                  filemode;
int                   status;

/* Access the file for reading */
strcpy(granuleID, "L1B_21_input.dat");
dataType = TK_L1B_21;
filemode = TK_NEW_FILE;
status = TKopen(granuleID, dataType, filemode, &granuleHandle1B21);

/* Check the Error Status */
if (status != TK_SUCCESS)
/* handle error processing here */

status = TKwriteHeader(&granuleHandle1b21, &L1b21header);
```

```
/* Check the Error Status */
if (status != TK_SUCCESS)
/* handle error processing here */
```

RETURN VALUES: TK_SUCCESS - Routine completed successfully.
W_TK_BADPIDWH - An invalid product ID was passed to TKwriteHeader
W_TK_BADFILEMODWH - An invalid file mode was passed to TKwriteHeader

PREREQUISITES: Before calling TKwriteHeader(), a file must be opened for reading by calling TKopen(). When the file is no longer needed, it should be closed by calling TKclose().

3.2 ROUTINE SPECIFICATION, FORTRAN VERSION

NAME: TKclose()

DESCRIPTION: This routine closes a data product file.

USAGE: #include "IO.h"

```
INTEGER FUNCTION TKclose(granuleHandle)
RECORD /WRAPPER_HANDLE/ granuleHandle
```

INPUTS: granuleHandle -
A structure containing information about the file which is to be closed. granuleHandle is returned by TKopen().

OUTPUTS: None.

DETAILS: Detailed descriptions of the input and output parameters and return codes can be found in the Appendix "Parameter Dictionary".

EXAMPLES:

```
#include "IO.h"
C       Include the I/O header file for TMI
        #include "IO_TMI.h"

C       Declare Variables
        RECORD /WRAPPER_HANDLE/ granuleHandle1B11
        INTEGER  status

        status = TKclose(granuleHandle1B11)

C       Check the Error Status
        IF (status .NE. TK_SUCCESS) THEN
C               handle error processing here
        ENDIF
```

RETURN VALUES:

TK_SUCCESS	- Routine completed successfully.
TK_FAIL	- Access routine failed.

PREREQUISITES: Before closing the file by TKclose(), the file should have been opened by TKopen().

NAME: TKopen()

DESCRIPTION: This routine opens a data product file prior to reading or writing product data or metadata.

USAGE: #include "IO.h"

```
INTEGER FUNCTION Tkopen (granuleID, dataType, filemode,
                                    granuleHandle)
CHARACTER*TK_FNAME_LEN granuleID
INTEGER                dataType
CHARACTER*1            filemode
RECORD /WRAPPER_HANDLE/        granuleHandle
```

INPUTS: granuleID -
A character string containing the name of the file to be opened. A maximum length of 255 characters is allowed.

dataType -
An unique identifier that specifies the type of data product being read, e.g., TK_L1B_11 for the 1B-11 algorithm product. A complete list of dataTypes can be found in the Parameter Dictionary.

filemode -
Access mode, TK_READ_ONLY and TK_NEW_FILE. TK_READ_ONLY opens the file in read only mode, without changing the metadata; and TK_NEW_FILE opens the file in read only mode but initializes the metadata with default values.

OUTPUTS: granuleHandle -
A structure passed to subsequent I/O routines. This structure is manipulated internally by TKopen() and other toolkit routines.

DETAILS: All files opened by TKopen() must be closed by or writing to a file. When a file is opened, the file pointer is positioned at the beginning of the orbit, not the beginning of the granule. The file pointer can be repositioned using TKseek().

Opening a file with mode=TK_NEW_FILE will initialize the metadata for that file with default values and allow write-only access to the file. These values can be changed using one of the metadata access routines.

A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

EXAMPLES:

```
C       Include the IO header file for TMI too.
        #INCLUDE "IO_TMI.h"

C       Declare Variables
        RECORD /WRAPPER_HANDLE/          granuleHandle1B11
        CHARACTER*TK_FNAME_LEN granuleID
        INTEGER                dataType
        CHARACTER*1            filemode
        INTEGER                status

C       Access the File for Reading
        dataType = TK_L1B_11
        filemode = TK_READ_ONLY
        status = TKopen(granuleID, dataType, filemode,
                        granuleHandle1B11)

C       Check the Error Status
        IF (status .NE. TK_SUCCESS) THEN
C           perform error handling here
        ENDIF
```

RETURN VALUES:

TK_SUCCESS	- Routine completed successfully.
TK_FAIL	- Open routine failed.

PREREQUISITES: None.

NAME: TKreadGrid()

DESCRIPTION: This routine reads gridded data from Level 3 TRMM satellite and L2 and L3 GV products.

USAGE:

```
#include "IO.h"

INTEGER FUNCTION TKreadGrid(granuleHandle,
                planetGrid)
RECORD /WRAPPER_HANDLE/ granuleHandle
RECORD /Planetary grid type/ planetGrid
```

INPUTS: granuleHandle -
A structure containing information about the file from which data can be obtained. granuleHandle is returned by TKopen().

OUTPUTS: planetGrid -
A structure containing the complete grid data which is obtained from the input data product. This structure must be declared to correspond to the data product being read.

DETAILS: A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

EXAMPLES:

```
#include "IO.h"
C       Include the I/O header file for TMI
#include "IO_TMI.h"

RECORD /L3A_11_PLANETGRID/      L3A11Grid
RECORD /WRAPPER_HANDLE/             granuleHandle3A11
CHARACTER*TK_FNAME_LEN          granuleID
INTEGER                         dataType
CHARACTER*1                     filemode
INTEGER                         status

C       Access the file for reading
granuleID =  'L3A_11_input.dat'
dataType = TK_L3A_11
filemode = TK_READ_ONLY
status = TKopen(granuleID, dataType, filemode,
                granuleHandle3A11)
```

```
      C     Check the Error Status
            IF (status .NE. TK_SUCCESS) THEN
      C     handle error processing here
            ENDIF
            status = TKreadGrid(granuleHandle3A11, l3A11Grid)

      C     Check the Error Status
            IF (status .NE. TK_SUCCESS)
      C     handle error processing here
```

RETURN VALUES: TK_SUCCESS - Routine completed successfully.
TK_FAIL - Routine failed.

PREREQUISITES: Before calling TKreadGrid(), a file must be opened for reading by calling TKopen(). When the file is no longer needed, it should be closed by calling TKclose().

NAME: TKreadlsm()

DESCRIPTION: Reads the land-sea database and returns a code that specifies if a given lat-lon point is, land, ocean, coast, ice, or coast next to ice.

USAGE:

```
#include "landsea.h"

INTEGER FUNCTION TKreadlsm(lat, lon)
REAL*4  LAT
REAL*4  LON
```

INPUTS:

lat -
Latitude coordinate in degrees. Latitude range is from -90 degrees to +90 degrees.

lon -
Longitude of the coordinate in degrees. Longitude range is from 0 to 359 degrees. Negative longitudes between -180 and 0 are also accepted.

OUTPUTS: None.

DETAILS: The data file must be located in the directory '$TSDISTK/data'. The data filename is 'dbglobe93.grd'.

EXAMPLES:

```
C       Include the Land-sea header file
        #include "landsea.h"

C       Declare Variables
        REAL*4      lat
        REAL*4      lon
        INTEGER*4   lstype

C       Read the land sea data file for a give latitude and
C       longitude

        lat = 23.0
        lon = 5.0
        lstype = TKreadlsm(lat, lon)
```

```
      IF (lstype .EQ. TK_LAND) THEN
C     do processing when over land
      ELSE IF (lstype .EQ. TK_OCEAN) THEN
C     do processing when over ocean
      ELSE IF (lstype .EQ. TK_COAST) THEN
C     do processing when along coast
      ENDIF
```

RETURN VALUES: The values returned are TK_LAND (land), TK_ICE (ice), TK_OCEAN (ocean), TK_COAST (coast), and TK_CICE (coast next to ice).

PREREQUISITES: None.

NAME: TKreadMetadataChar()

DESCRIPTION: This routine reads individual character metadata elements from an HDF data product.

USAGE:

```
#include "IO.h"

INTEGER FUNCTION TKreadMetadataChar(granuleHandle,
        parameter,value)
RECORD /WRAPPER_HANDLE/ granuleHandle
CHARACTER*(*)  parameter
CHARACTER*(*)  value
```

INPUTS: granuleHandle -
A structure containing information about the data product from which the metadata is read. granuleHandle is returned by TKopen().

parameter -
A string containing the name of the metadata element to be read from the data product.

OUTPUTS: value -
The character value corresponding to the metadata element specified by parameter.

DETAILS: All metadata is stored internally in character format. The metadata access routines translate the character data to the appropriate format as needed. A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

EXAMPLES:

C

```
#INCLUDE "IO.h"
Include the instrument specific IO file for TMI.
#INCLUDE "IO_TMI.h"

RECORD /WRAPPER_HANDLE/          granuleHandle1B11
CHARACTER*TK_FNAME_LEN granuleID
INTEGER                dataType
CHARACTER*1            filemode
INTEGER                status
CHARACTER*TK_ALGID_LEN algID
```

```
C       Access the file for reading.
        granuleID= 'L1B_11_input.dat'
        dataType = TK_L1B_11
        filemode = TK_READ_ONLY
        status = TKopen(granuleID, dataType, filemode,
                granuleHandle1B11)

C       Check the Error Status
        IF (status .NE. TK_SUCCESS) THEN
C       perform error handling here
        ENDIF

        status = TKreadMetadataChar (granuleHandle1B11,
                TK_ALGORITHM_ID, algID)

C       Check the Error Status
        IF (status .NE.  TK_SUCCESS) THEN
C       perform error handling here
        ENDIF
```

RETURN VALUES:

TK_SUCCESS	- Routine completed successfully.
TK_FAIL	- Access routine failed.

PREREQUISITES: Before calling TKreadMetadataChar(), a data product must be opened for reading by calling TKopen(). When the data product is no longer needed, it should be closed by calling TKclose().

NAME: TKreadMetadataFloat()

DESCRIPTION: This routine reads floating point metadata elements from an HDF data product.

USAGE: #include "IO.h"

```
INTEGER FUNCTION TKreadMetadataFloat(granuleHandle,
                parameter,value)
RECORD /WRAPPER_HANDLE/ granuleHandle
CHARACTER*(*)  parameter
REAL value
```

INPUTS: granuleHandle -
A structure containing information about the data product from which metadata is retrieved. granuleHandle is returned by TKopen().

parameter -
A string containing the name of the metadata element to be read from the data product.

OUTPUTS: value -
A float value obtained from the metadata element specified by 'parameter'.

DETAILS: All metadata is stored internally in character format. The metadata access routines translate the character data to the appropriate format as needed. Detailed descriptions of the input and output parameters and return codes can be found in the "Parameter Dictionary".

EXAMPLES:

C

```
#INCLUDE "IO.h"
Include the instrument specific IO file for TMI.
#INCLUDE "IO_TMI.h"

RECORD /WRAPPER_HANDLE/          granuleHandle1B11
CHARACTER*TK_FNAME_LEN granuleID
INTEGER                dataType
CHARACTER*1            filemode
INTEGER                status
REAL                   percentBadMissPix
```

```
      C       Access the file
              granuleID='L1B_11_input.dat'
              dataType = TK_L1B_11
              filemode = TK_READ_ONLY
              status = TKopen(granuleID, dataType, filemode,
                     granuleHandle1B11)

      C       Check the Error Status
              IF (status .NE. TK_SUCCESS) THEN
      C       perform error handling here
              ENDIF

              status = TKreadMetadataFloat(granuleHandle1B11,
              TK_PERCENT_BAD_MISS_PIXEL,
              percentBadMissPix)

      C       Check the Error Status
              IF (status .NE. TK_SUCCESS) THEN
      C       perform error handling here
              ENDIF
```

RETURN VALUES:

TK_SUCCESS	- Routine completed successfully.
TK_FAIL	- Routine failed.

PREREQUISITES: Before calling TKreadMetadataFloat(), a data product must be opened for reading by calling TKopen(). When the data product is no longer needed, it should be closed by calling TKclose().

NAME: TKreadMetadataInt()

DESCRIPTION: This routine reads individual integer metadata elements from an HDF data product.

USAGE:

```
#include "IO.h"

INTEGER FUNCTION TKreadMetadataInt(granuleHandle,
        parameter,value)
RECORD /WRAPPER_HANDLE/ granuleHandle
CHARACTER*(*)  parameter
INTEGER  value
```

INPUTS: granuleHandle -
A structure containing information about the data product which data is being read. granuleHandle is returned by TKopen().

parameter -
A string containing the name of the metadata element to be read from the data product.

OUTPUTS: value -
The character value corresponding to the metadata element specified by 'parameter'.

DETAILS: All metadata is stored internally in character format. The metadata access routines translate the character data to the appropriate format as needed. Detailed descriptions of the input and output parameters and return codes can be found in the "Parameter Dictionary".

EXAMPLES:

C

```
#INCLUDE "IO.h"
Include the instrument specific IO file for TMI.
#INCLUDE "IO_TMI.h"

RECORD /WRAPPER_HANDLE/          granuleHandle1B11
CHARACTER*TK_FNAME_LEN granuleID
INTEGER                dataType
CHARACTER*1            filemode
INTEGER                status
INTEGER                orbitNum
```

```
      C       Access the file for reading
              granuleID='L1B_11_input.dat'
              dataType = TK_L1B_11
              filemode = TK_READ_ONLY
              status = TKopen(granuleID, dataType, filemode,
                     granuleHandle1B11)

      C       Check the Error Status
              IF (status .NE. TK_SUCCESS) THEN
      C       perform error handling here
              ENDIF

              status = TKreadMetadataInt(granuleHandle1B11,
              TK_ORBIT_NUMBER, orbitNum)

      C       Check the Error Status
              IF (status .NE. TK_SUCCESS) THEN
      C       perform error handling here
              ENDIF
```

RETURN VALUES:

TK_SUCCESS	- Routine completed successfully.
TK_FAIL	- Open routine failed.

PREREQUISITES: Before calling TKreadMetadataInt(), a data product must be opened for reading by calling TKopen(). When the data product is no longer needed, it should be closed by calling TKclose().

NAME: TKreadScan()

DESCRIPTION: This routine reads scan based satellite data from a data product and store them into the product data structure.

USAGE:

```
#include "IO.h"

INTEGER FUNCTION TKreadScan(granuleHandle,
      swathData)
RECORD /WRAPPER_HANDLE/ granuleHandle
C     'scanline data' should be replaced with a specific
C     structure name.
RECORD /scanline data/ swathData
```

INPUTS: granuleHandle -
A structure containing information about the data product from which data is read. granuleHandle is returned by TKopen().

OUTPUTS: swathData -
A structure containing the complete scanline data which is obtained from the input data product. This structure must be declared to correspond to the data product being read.

DETAILS: TKopen positions the file pointer at the beginning of the orbit, not the granule. Thus, the first call to readScan returns the first scan in the orbit. The scan number is updated on each call so consecutive calls return consecutive scans. TKseek() may be used to move forward or backward by a certain number of scans.

Detailed descriptions of the input and output parameters and return codes can be found in the Appendix "Parameter Dictionary".

EXAMPLES:

```
#INCLUDE "IO.h"
C     Include the instrument specific IO file for TMI.
#INCLUDE "IO_TMI.h"

RECORD /L1B_11_SWATHDATA/      L1B11Data
RECORD /WRAPPER_HANDLE/            granuleHandle1B11
CHARACTER*TK_FNAME_LEN         granuleID
INTEGER                        dataType
CHARACTER*1                    filemode
INTEGER                        status
```

```
      C       Access the file
              granuleID='L1B_11_input.dat'
              dataType = TK_L1B_11
              filemode = TK_READ_ONLY
              status = TKopen(granuleID,dataType, filemode,
                     granuleHandle1B11)

      C       Check the Error Status
              IF (status .NE. TK_SUCCESS) THEN
      C       handle error processing here
              ENDIF

              status = TKreadScan(granuleHandle1B11,
                     L1B11Data)

      C       Check the Error Status
              IF (status .NE. TK_SUCCESS) THEN
      C       handle error processing here
              ENDIF
```

RETURN VALUES:

TK_SUCCESS	- Routine completed successfully.
TK_FAIL	- Routine failed.

PREREQUISITES: Before calling TKreadScan(), a data product must be opened for reading by calling TKopen(). When the data product is no longer needed, it should be closed by calling TKclose().

NAME: TKreadTopo()

DESCRIPTION: This routine reads the ETOPO5 database, returning an elevation in meters for a given coordinate point.

USAGE:

```
#include "etopo5.h"

INTEGER*2 FUNCTION TKreadTopo(lat, lon)
REAL*4 LAT
REAL*4 LON
```

INPUTS:

lat -
Latitude coordinate, in degrees, of the point for which elevation is desired. Latitude runs from -90 degrees to +90 degrees.

lon -
Longitude of the coordinate, in degrees, of the point for which elevation is desired. Longitude runs from 0 to 359 degrees. Negative longitudes between -180 and 0 are also accepted.

OUTPUTS: None.

DETAILS: The ETOPO5 database is gridded in 1/12 degree bins. The entire file is read in each time the routine is called. The data file must reside in '$TSDISTK/data/etop05.dat'.

EXAMPLES:

```
C     Include the topo header file
      #include "etopo5.h"

C     Declare Variables
      REAL*4        lat, lon
      INTEGER*2     elevation

C     Read the ETOPO5 data file for a give latitude and
C     longitude

      lat = 50.0
      lon = 17.0
      elevation = TKreadTopo(lat, lon)

C     Do processing
```

RETURN VALUES: Returns the elevation of the coordinate point in meters. When over the ocean, the elevation may be negative.

PREREQUISITES: None.

NAME: TKreportError()

DESCRIPTION: This routine records an error message based on the error number provided by the user or the status returned by a routine.

USAGE: #include "TS_XX_YY.h"

SUBROUTINE TKreportError(errorNumber)

INTEGER errorNumber

INPUTS: errorNumber -
A parameter that is used to identify particular error message. Each error number corresponds to a specific error message.

OUTPUTS: None.

DETAILS: The complete list of error codes can be found in the Parameter Dictionary.

When TKreportError is called by the user, it expands as a macro to a different routine that includes '__LINE__' and '__FILE__' after the errorNumber. These two additional arguments capture the line number and file name of the call to TKreportError and are printed to the console and the log file along with the error message.

The error messages are in files named like "TS_TK_15.h", where TS stands for TSDIS, TK stands for Toolkit, and 15 means that all the error numbers start at 15000. The appropriate include file must be #included in the source code you write for the error routines to workcorrectly.

The routine then looks for the error messages in the file "TS_15", which must be located in the directory $TSDISTK/include.

EXAMPLES: This example shows the preprocessing steps, the variable declaration, and a call to TKreportError in two different conditions. The parameter in the call to TKreportError is the error number.

```
C       Include Files
        #include "TS_XX_YY.h"

C       Declare Variables
        INTEGER     status
        INTEGER     nr
```

```
C       Call a processing routine that returns an error
C       status.  The error status can be a success or be a
C       fatal error.

        status = process_data()

C       Check if the status is a success, if not a success
C       then report the fatal error.  E_TK_INVALID_DATA is
C       a TSU defined message that has been registered with
C       TSDIS.

        IF (status .NE. TK_SUCCESS)
        CALL TKreportError(E_TK_INVALID_DATA)

        ...

C       call a user defined processing routine that returns the number of records
        read.

        nr = read_records()

C       If the number of records read is less than the
C       expected number of records, NUM_REC, then report
C       the error E_TK_READ_FAIL.  Both the warning message
C       and the error message, E_TK_READ_FAIL, are TSU
C       defined messages.  Any messages that are reported
C       with TKreportError should be registered with TSDIS.


        IF (nr .LT. NUM_REC)
        CALL TKreportError(E_TK_READ_FAIL)
```

RETURN VALUES: None.

PREREQUISITES: None.

NAME: TKreportWarning()

DESCRIPTION: This routine reports a warning message based on the warning number provided by the user, or a status returned by a routine.

USAGE: #include "TS_XX_YY.h"

SUBROUTINE TKreportWarning(warnNumber)

INTEGER warnNumber

INPUTS: warnNumber -
A parameter that is used to index a particular warning message. Each warning number corresponds to a specific message.

OUTPUTS: None.

DETAILS: The complete list of warning codes can be found in the Parameter Dictionary.

The routine records the corresponding warning message in a log file, prints the message to the console, and returns control to the calling program. Compare with TKreportError().

When TKreportWarning is called by the user, it expands as a macro to a different routine that includes '__LINE__' and '__FILE__' after the warnNumber. These two additional arguments capture the line number and file name of the call to TKreportWarning and are printed to the operator and the log file along with the error message.

The error messages are in files named like "TS_TK_15.h", where TS stands for TSDIS, TK stands for Toolkit, and 15 means that all the error numbers start at 15000. The appropriate include file must be #included in the source code you write for the error routines to workcorrectly.

The routine then looks for the error messages in the file "TS_15", which must be located in the directory $TSDISTK/include.

EXAMPLES: This example shows the preprocessing steps, the variable declaration, and a call to TKreportWarning with two different conditions. The parameter in the call to TKreportWarning is the warning number.

```
C       Include files
        #include "TS_XX_YY.h"

C       Declare Variables
        INTEGER     status
        INTEGER     nr

C       Call  a processing routine that returns an error
C       status.

        status = process_data()

C       Check if the status is a success, if not a success
C       then report the warning error.  W_TK_INVALID_DATA
C       is a TSU defined message that has been registerd
C       with TSDIS.

        IF (status .NE. TK_SUCCESS)
        CALL TKreportWarning(W_TK_INVALID_DATA)

        ...

C       Call a user defined processing routine that returns
C       the number of records read.  nr = read_records ()

C       If the number of records read is less than the
C       expected number of records, NUM_REC, then report
C       the warning W_TK_READ_FAIL.  The warning message,
C       W_TK_READ_FAIL, are TSU defined messages.  Any
C       messages that are reported with TKreportError
C       should be registered with TSDIS.

        IF (nr .LT. NUM_REC)
        CALL TKreportWarning(W_TK_READ_FAILED)
```

RETURN VALUES: None.

PREREQUISITES: None.

NAME: TKseek()

DESCRIPTION: This routine is used to move the file pointer to a specified position within the file; this enables reading of an specified scan line.

USAGE:
```
#include "IO.h"
INTEGER FUNCTION TKseek(granuleHandle, offset, type)
RECORD /WRAPPER_HANDLE/ granuleHandle
INTEGER offset
INTEGER type
```

INPUTS: granuleHandle -
A structure containing information about an open file. granuleHandle is returned by TKopen().

offset -
The specification of this parameter depends on the value of the 'type' parameter.

If type = TK_REL_SCAN_OFF, offset is an integer specifying the number of scan lines to move, relative to the current scan. If offset=5, this would move the file pointer forward by 5 scans. If offset=-1 the file pointer would move back by 1 scan.

If type = TK_ABS_SCAN_OFF, offset is an integer specifying the scan line relative to the beginning of the file.

type -
The type parameter will be one of two values. If TK_REL_SCAN_OFF is used as the type parameter, then the seek is relative to the current scan at the file pointer. If TK_ABS_SCAN_OFF is used, then the seek is from the beginning of the data product.

OUTPUTS: None

DETAILS: Detailed descriptions of the input and output parameters and return codes can be found in the Appendix "Parameter Dictionary".

EXAMPLES:

C Include the instrument specific IO file for TMI.
```
#include "IO.h"
#include "IO_TMI.h"
```

```
      RECORD /WRAPPER_HANDLE/          granuleHandle1B11
      CHARACTER*TK_FNAME_LEN granuleID
      INTEGER                dataType
      CHARACTER*1            filemode
      INTEGER                offset
      INTEGER                status

C     Access the file for reading
      granuleID='L1B_11_input.dat'
      dataType = TK_L1B_11
      filemode = TK_READ_ONLY
      status = TKopen(granuleID, dataType, filemode,
     &         granuleHandle1B11)

C     Check the Error Status
      IF (status .NE. TK_SUCCESS) THEN
C     handle error processing here
      ENDIF

C     Reset the file pointer to the beginning of the Granule
      offset = 5
      status = TKseek(granuleHandle1B11, offset, TK_ABS_SCAN_OFF)

C     Check the Error Status
      IF (status .NE. TK_SUCCESS) THEN
C      handle error processing here
      ENDIF
```

RETURN VALUES: If successful, the routine returns the offset. If unsuccessful, the actual offset, in the units requested, will be returned. Thus, if the file pointer was positioned 5 scans from the end of file, and an offset of 10 scans was requested, the routine will return the value 5.

PREREQUISITES: Before using TKseek(), the file must have been opened by TKopen().

NAME: TKwriteGrid()

DESCRIPTION: This routine writes gridded data to Level 3 TRMM satellite and Level 2 and Level 3 GV data products.

USAGE: #include "IO.h"

```
INTEGER FUNCTION TKwriteGrid(granuleHandle, planetGrid)
RECORD /WRAPPER_HANDLE/ granuleHandle
RECORD /Planetary grid type/ planetGrid
```

C 'Planetary grid type' should be replaced with a specific structure type name.

INPUTS: granuleHandle -
A structure containing information about the file from which data can be obtained. granuleHandle is returned by TKopen().

planetGrid -
A structure containing the complete grid data which is obtained from the input file. This structure must be declared to correspond to the data file being read.

OUTPUTS: None.

DETAILS: A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

EXAMPLES: #include "IO.h"
C Include the I/O header file for TMI
#include "IO_TMI.h"

```
RECORD /L3A_11_PLANETGRID/      L3A11Grid
RECORD /WRAPPER_HANDLE/             granuleHandle3A11
CHARACTER*TK_FNAME_LEN          granuleID
INTEGER                         dataType
CHARACTER*1                     filemode
INTEGER                         status
```

```
      C       Access the file for writing
              granuleID =  'L3A_11_output.dat'
              dataType = TK_L3A_11
              filemode = TK_NEW_FILE
              status = TKopen(granuleID, dataType, filemode,
                     granuleHandle3A11)

      C       Check the Error Status
              IF (status .NE. TK_SUCCESS) THEN
      C       handle error processing here
              ENDIF
              status = TKwriteGrid(granuleHandle3A11, l3A11Grid)

      C       Check the Error Status
              IF (status .NE. TK_SUCCESS)
      C       handle error processing here
```

RETURN VALUES:

TK_SUCCESS	- Routine completed successfully.
TK_FAIL	- Routine failed.

PREREQUISITES: Before calling TKreadGrid(), a data product must be opened for reading by calling TKopen(). When the data product is no longer needed, it should be closed by calling TKclose().

NAME: TKwriteMetadataChar()

DESCRIPTION: This routine writes individual character metadata items to an HDF data product.

USAGE: #include "IO.h"

```
INTEGER FUNCTION TKwriteMetadataChar(granuleHandle,
                parameter,value)
RECORD /WRAPPER_HANDLE/ granuleHandle
CHARACTER*(*)  parameter
CHARACTER*(*)  value
```

INPUTS: ranuleHandle -
A structure containing information about the file to which metadata information can be written. granuleHandle is returned by TKopen().

parameter -
A string containing the name of the metadata element to be read.

OUTPUTS: value -
The character value corresponding to the metadata element specified by parameter.

DETAILS: All metadata is stored internally in character format. The metadata access routines translate the character data to the appropriate format as needed. Detailed descriptions of the input and output parameters and return codes can be found in the Appendix "Parameter Dictionary".

EXAMPLES:

C

```
#INCLUDE "IO.h"
Include the IO header file for TMI
#INCLUDE "IO_TMI.h"

RECORD /WRAPPER_HANDLE/          granuleHandle1B11
CHARACTER*TK_FNAME_LEN granuleID
INTEGER                dataType
CHARACTER*1            filemode
INTEGER                status
CHARACTER*TK_ALGID_LEN algID
```

```
      C       Access the file for writing
              granuleID='L1B_11_input.dat'
              dataType = TK_L1B_11
              filemode = TK_NEW_FILE
              status = TKopen(granuleID, dataType, filemode,
                     granuleHandle1B11)

      C       Check the Error Status
              IF (status .NE. TK_SUCCESS) THEN
      C         handle error processing here
              ENDIF

              algID = 'TK_L1B_11'

              status = TKwriteMetadataChar(granuleHandle1B11,
                     TK_ALGORITHM_ID, algID)

      C       Check the Error Status
              IF (status .NE. TK_SUCCESS) THEN
      C       handle error processing here
              ENDIF
```

RETURN VALUES:

TK_SUCCESS	- Routine completed successfully.
TK_FAIL	- Routine failed.

PREREQUISITES: Before calling TKwriteMetadataChar(), a data product must be opened by calling TKopen(). When the data product is no longer necessary, it should be closed by calling TKclose().

NAME: TKwriteMetadataFloat()

DESCRIPTION: This routine writes individual floating point metadata elements to an HDF data product.

USAGE:

```
#include "IO.h"

INTEGER FUNCTION TKwriteMetadataFloat(granuleHandle,
                                      parameter,value)
RECORD /WRAPPER_HANDLE/ granuleHandle
CHARACTER*(*)  parameter
REAL value
```

INPUTS:

granuleHandle -
A structure containing information about the data product to which metadata information can be written. granuleHandle is returned by TKopen().

parameter -
A string containing the name of the metadata element to be retrieved from the data product.

OUTPUTS:

value -
The character value corresponding to the metadata element specified by parameter.

DETAILS: All metadata is stored internally in character format. The metadata access routines translate the character data to the appropriate format as needed. Detailed description of the input and output parameters and return codes can be found in the Appendix "Parameter Dictionary".

EXAMPLES:

C

```
#INCLUDE "IO.h"
Include the instrument specific IO file for TMI.
#INCLUDE "IO_TMI.h"

RECORD /WRAPPER_HANDLE/   granuleHandle1B11
CHARACTER*TK_FNAME_LEN granuleID
INTEGER                   dataType
CHARACTER*1               filemode
INTEGER                   status
REAL                      percentBadMissPix
```

```
      C       Access the file for writing
              granuleID='L1B_11_output.dat'
              dataType = TK_L1B_11
              filemode = TK_NEW_FILE
              status = TKopen(granuleID, dataType, filemode,
                     granuleHandle1B11)

      C       Check the Error Status
              IF (status .NE. TK_SUCCESS) THEN
      C       handle error processing here
              ENDIF

              percentBadMissPix = 20.3
              status = TKwriteMetadataFloat(granuleHandle1B11,
                     TK_PERCENT_BAD_MISS_PIXEL,
                     percentBadMissPix)

      C       Check the Error Status
              IF (status .NE. TK_SUCCESS) THEN
      C       handle error processing here
              ENDIF
```

RETURN VALUES:

TK_SUCCESS	- Routine completed successfully.
TK_FAIL	- Routine failed.

PREREQUISITES: Before calling TKwriteMetadataFloat(), a data product must be opened for writing by calling TKopen(). When the data product is no longer needed, it should be closed by calling TKclose().

NAME: TKwriteMetadataInt()

DESCRIPTION: This routine writes individual integer metadata items to an HDF data product.

USAGE: #include "IO.h"

```
INTEGER FUNCTION TKwriteMetadataInt(granuleHandle,
        parameter,value)
RECORD /WRAPPER_HANDLE/ granuleHandle
CHARACTER*(*)  parameter
INTEGER  value
```

INPUTS: granuleHandle -
A structure containing information about the data product to which metadata information is being written. granuleHandle is returned by TKopen().

parameter -
A string which containing the name of the metadata element to be written to the data product.

OUTPUTS: value -
The character value corresponding to the metadata element specified by parameter.

DETAILS: All metadata is stored internally in character format. The metadata access routines translate the character data to the appropriate format as needed. Detailed description of the input and output parameters and return codes can be found in the Appendix "Parameter Dictionary".

EXAMPLES:

C

```
#INCLUDE "IO.h"
Include the instrument specific IO file for TMI.
 #INCLUDE "IO_TMI.h"

RECORD /WRAPPER_HANDLE/          granuleHandle2A12
CHARACTER*TK_FNAME_LEN granuleID
INTEGER                dataType
CHARACTER*1            filemode
INTEGER                status
INTEGER                orbitNum
```

```
C       Access the file for writing
        granuleID='L2A_12_output.dat'
        dataType = TK_L2A_12
        filemode = TK_NEW_FILE
        status = TKopen(granuleID, dataType, filemode,
                granuleHandle2A12)

C       Check the Error Status
        IF (status .NE.  TK_SUCCESS) THEN
C       handle error processing here
        ENDIF

        orbitNum = 104

        status = TKwriteMetadataInt(granuleHandle2A21,
                        TK_ORBIT_NUMBER,
                        orbitNum)

C       Check the Error Status
        IF (status .NE. TK_SUCCESS) THEN
C       handle error processing here
        ENDIF
```

RETURN VALUES:

TK_SUCCESS	- Routine completed successfully.
TK_FAIL	- Routine failed.

PREREQUISITES: Before calling TKwriteMetadataInt(), a data product must be opened for writing by calling TKopen(). When the data product is no longer needed, it should be closed by calling TKclose().

NAME: TKwriteScan()

DESCRIPTION: This routine writes scan based satellite product data to a data product.

USAGE:

```
#include "IO.h"

INTEGER FUNCTION TKwriteScan(granuleHandle,
                             swathData)
RECORD /WRAPPER_HANDLE/ granuleHandle
RECORD /scanline data/ swathData
```

INPUTS:

granuleHandle -
A structure containing information about the data product to which data can be written. granuleHandle is returned by TKopen ().

swathData -
A structure containing the complete scanline data which is to be written to the output data product. This structure must be declared to correspond to the data product being written.

OUTPUTS: None.

DETAILS: Each call to TKwriteScan increments the scan line number by one, so the output file is appended with each scan written.

Detailed description of the input and output parameters and return codes can be found in the Appendix "Parameter Dictionary".

EXAMPLES:

C

```
#INCLUDE "IO.h"
Include the I/O header file for TMI
#INCLUDE "IO_TMI.h"

RECORD /L1B_11_DATA/ l1B11Data
RECORD /WRAPPER_HANDLE/          granuleHandle1B11
CHARACTER*TK_FNAME_LEN granuleID
INTEGER                dataType
CHARACTER*1            filemode
INTEGER                status
```

```
C       Open the file for writing
        granuleID='L1B_11_input.dat'
        dataType = TK_L1B_11
        filemode = TK_NEW_FILE
        status = TKopen(granuleID, dataType, filemode,
             granuleHandle1B11)

C       Check the Error Status
        IF (status .NE. TK_SUCCESS) THEN
C       Perform error handling here
        ENDIF

        status = TKwriteScan(granuleHandle1B11,
             l1B11ScanData)

C       Check the Error Status
        IF (status .NE. TK_SUCCESS) THEN
C       perform error handling here
        ENDIF
```

RETURN VALUES:

TK_SUCCESS	- Routine completed successfully.
TK_FAIL	- Access routine failed.

PREREQUISITES: Before calling TKwriteScan(), a data product must be opened by calling TKopen(). When the data product is no longer needed, it should be closed by calling TKclose().

NAME: TKgetNvos()

DESCRIPTION: This routine returns the number of Volume Scans in the GV granule.

USAGE:
```
#include "IO.h"
#include "IO_GV.h"

INTEGER FUNCTION TKgetNvos(granuleHandle)
RECORD /WRAPPER_HANDLE/ granuleHandle
```

INPUTS: granuleHandle -
A structure containing information about the data product to which data can be written. granuleHandle is returned by TKopen().

OUTPUTS: None.

DETAILS: A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

EXAMPLES:
```
      C     IO.h contains the function prototypes for the
      C     majority of the I/O routines, including the GV
      C     read and write routines.

            #include "IO.h"

      C     Include the GV structure definitions and macros
            #include "IO_GV.h"

            RECORD /L1B_1C_GV/              l1BGVData
            RECORD /WRAPPER_HANDLE/             granuleHandle1BGV
            CHARACTER*TK_FNAME_LEN granuleID
            INTEGER                         dataType
            CHARACTER*1                     filemode
            INTEGER                         status
            INTEGER                         nvos

      C     Open the file for reading
            granuleID = 'L1B_GV_input.dat'
            dataType = TK_L1B_GV
            filemode = TK_READ_ONLY
            status = TKopen(granuleID, dataType, filemode,
                granuleHandle1BGV)
```

```
      C     Check the Error Status
            IF (status .NE. TK_SUCCESS) THEN
      C     Perform error handling here

            nvos = TKgetNvos(granuleHandle1BGV)

            IF (nvos .LE. 0) THEN
      C     Perform error handling here
```

RETURN VALUES: The number of volume scans is returned as an integer, or TK_FAIL if the routine failed.

PREREQUISITES: The file associated with granuleHandle must have been previously opened by TKopen().

NAME: TKgetNsensor()

DESCRIPTION: This routine returns the number of sensors in a GV VOS.

USAGE:
```
#include "IO.h"
#include "IO_GV.h""
INTEGER FUNCTION TKgetNsensor(granuleHandle)
RECORD /WRAPPER_HANDLE/ granuleHandle
```

INPUTS: granuleHandle -
A structure containing information about the data product to which data can be written. granuleHandle is returned by TKopen().

OUTPUTS: None.

DETAILS: A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

EXAMPLES:

```
      C     IO.h contains the function prototypes for the
      C     majority of the I/O routines, including the GV
      C     read and write routines.

            #include "IO.h"

      C     Include the GV structure definitions and macros
            #include "IO_GV.h"

            RECORD /L1B_1C_GV/              l1BGVData
            RECORD /WRAPPER_HANDLE/             granuleHandle1BGV
            CHARACTER*TK_FNAME_LEN granuleID
            INTEGER                         dataType
            CHARACTER*1                     filemode
            INTEGER                         status
            INTEGER                         nsensor

      C     Open the file for reading
            granuleID = 'L1B_GV_input.dat'
            dataType = TK_L1B_GV
            filemode = TK_READ_ONLY
            status = TKopen(granuleID, dataType, filemode,
                granuleHandle1BGV)
```

```
      C     Check the Error Status
            IF (status .NE. TK_SUCCESS) THEN
      C     Perform error handling here

            nsensor = TKgetNsensor(granuleHandle1BGV)
```

RETURN VALUES: Returns number of sensors if successful, or TK_FAIL if the routine failed.

PREREQUISITES: The file corresponding to the granuleHandle must have been opened previously by TKopen().

NAME: TKgetNparam()

DESCRIPTION: This routine returns the number of parameters in a GV volume scan.

USAGE:
```
#include "IO.h"
#include "IO_GV.h"

INTEGER FUNCTION TKgetNparam(granuleHandle,
                             VOSnumber)
RECORD /WRAPPER_HANDLE/granuleHandle
INTEGER  VOSnumber
```

INPUTS: granuleHandle -
A structure containing information about the file to which data can be written. granuleHandle is returned by TKopen().

VOSnumber -
The volume scan number. This is a sequential number from 0 to the number of VOS in the granule.

OUTPUTS: None.

DETAILS: A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

EXAMPLES:
```
      C     IO.h contains the function prototypes for the
      C     majority of the I/O routines, including the GV
      C     read and write routines.

            #include "IO.h"

      C     Include the GV structure definitions and macros
            #include "IO_GV.h"

            RECORD /L1B_1C_GV/              l1BGVData
            RECORD /WRAPPER_HANDLE/             granuleHandle1BGV
            CHARACTER*TK_FNAME_LEN granuleID
            INTEGER                         dataType
            CHARACTER*1                     filemode
            INTEGER                         status
            INTEGER                         nparam
            INTEGER                         nvos
```

```
C       Open the file for reading
        granuleID = 'L1B_GV_input.dat'
        dataType = TK_L1B_GV
        filemode = TK_READ_ONLY
        status = TKopen(granuleID, dataType, filemode,
                granuleHandle1BGV)

C       Check the Error Status
        IF (status .NE. TK_SUCCESS) THEN
C       Perform error handling here

        nvos = 5
        nparam = TKgetNparam(granuleHandle1BGV, nvos)
```

RETURN VALUES: Returns the number of parameters in the VOS if successful. Returns TK_FAIL if unsuccessful.

PREREQUISITES: The file corresponding to the granuleHandle must have been opened previously by TKopen().

NAME: TKgetNcell()

DESCRIPTION: This routine returns the number of cells for a given volume scan number and parameter number.

USAGE:
```
#include "IO.h"
#include "IO_GV.h"
INTEGER FUNCTION TKgetNcell(granuleHandle,VOSnum,
        paramNum)
RECORD /WRAPPER_HANDLE/granuleHandle
INTEGER  VOSnum
INTEGER  paramNum
```

INPUTS: granuleHandle -
A structure containing information about the file to which data can be written. granuleHandle is returned by TKopen().

VOSnum -
The volume scan number. This is a sequential number from 0 to the number of VOSs in the granule.

paramNum -
The parameter number for the give volume scan.

OUTPUTS: None.

DETAILS: A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

EXAMPLES:
```
      C     IO.h contains the function prototypes for the
      C     majority of the I/O routines, including the GV
      C     read and write routines.

            #include "IO.h"

      C     Include the GV structure definitions and macros
            #include "IO_GV.h"

            RECORD /L1B_1C_GV/              l1BGVData
            RECORD /WRAPPER_HANDLE/         granuleHandle1BGV
            CHARACTER*TK_FNAME_LEN granuleID
            INTEGER                         dataType
```

```
      CHARACTER*1             filemode
      INTEGER                 status
      INTEGER                 ncell
      INTEGER                 vosNum
      INTEGER                 paramNum

C     Open the file for reading
      granuleID = 'L1B_GV_input.dat'
      dataType = TK_L1B_GV
      filemode = TK_READ_ONLY
      status = TKopen(granuleID, dataType, filemode,
     &          granuleHandle1BGV)

C     Check the Error Status
      IF (status .NE. TK_SUCCESS) THEN
C       Perform error handling here

      vosNum = 2
      paramNum = 2
      ncell = TKgetNcell(granuleHandle1BGV, vosNum, paramNum )
```

RETURN VALUES: Returns the number of Cells in the VOS. TK_FAIL - Routine Failed.

PREREQUISITES: File corresponding to granuleHandle must have been opened previously by calling TKopen().

NAME: TKgetNray()

DESCRIPTION: This routine returns the number of rays in a specified VOS.

USAGE:
```
#include "IO.h"
#include "IO_GV.h"
INTEGER FUNCTION TKgetNray(granuleHandle, VOSnum)
RECORD /WRAPPER_HANDLE/ granuleHandle
INTEGER      VOSnum
```

INPUTS: granuleHandle -
A structure containing information about the file to which data can be written. granuleHandle is returned by TKopen().

VOSnum -
The volume scan number. This is a sequential number from 0 to the number of VOSs in the granule.

OUTPUTS: None.

DETAILS: A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

EXAMPLES:

```
C     IO.h contains the function prototypes for the
C     majority of the I/O routines, including the GV
C     read and write routines.

      #include "IO.h"

C     Include the GV structure definitions and macros
      #include "IO_GV.h"

      RECORD /L1B_1C_GV/               l1BGVData
      RECORD /WRAPPER_HANDLE/              granuleHandle1BGV
      CHARACTER*TK_FNAME_LEN granuleID
      INTEGER                dataType
      CHARACTER*1            filemode
      INTEGER                status
      INTEGER                nray
      INTEGER                vosNum
```

```
C       Open the file for reading
        granuleID = 'L1B_GV_input.dat'
        dataType = TK_L1B_GV
        filemode = TK_READ_ONLY
        status = TKopen(granuleID, dataType, filemode,
             granuleHandle1BGV)

C       Check the Error Status
        IF (status .NE. TK_SUCCESS) THEN
C       Perform error handling here

        vosNum = 5
        nray = TKgetNray(granuleHandle1BGV, vosNum )
```

RETURN VALUES: Returns the number of rays in the VOS, or TK_FAIL if unsuccessful.

PREREQUISITES: File corresponding to granuleHandle must have been opened previously by calling TKopen().

NAME: TKgetNsweep()

DESCRIPTION: This routine returns the number of sweeps for a given Volume Scan.

USAGE:
```
#include "IO.h"
#include "IO_GV.h"
INTEGER FUNCTION TKgetNsweep(granuleHandle, VOSnum)
RECORD /WRAPPER_HANDLE/  granuleHandle
INTEGER  VOSnum
```

INPUTS: granuleHandle -
A structure containing information about the file to which data can be written. granuleHandle is returned by TKopen().

VOSnum -
The volume scan number. This is a sequential number between 0 and the number of VOSs in the granule.

OUTPUTS: None.

DETAILS: A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

EXAMPLES:

```
C       IO.h contains the function prototypes for the
C       majority of the I/O routines, including the GV
C       read and write routines.

        #include "IO.h"

C       Include the GV structure definitions and macros
        #include "IO_GV.h"

        RECORD /L1B_1C_GV/              l1BGVData
        RECORD /WRAPPER_HANDLE/             granuleHandle1BGV
        CHARACTER*TK_FNAME_LEN granuleID
        INTEGER                         dataType
        CHARACTER*1                     filemode
        INTEGER                         status
        INTEGER                         nsweep
        INTEGER                         vosNum
```

```
C       Open the file for reading
        granuleID = 'L1B_GV_input.dat'
        dataType = TK_L1B_GV
        filemode = TK_READ_ONLY
        status = TKopen(granuleID, dataType, filemode,
                granuleHandle1BGV)

C       Check the Error Status
        IF (status .NE. TK_SUCCESS) THEN
C         Perform error handling here

        vosNum = 2
        nsweep = TKgetNsweep(granuleHandle1BGV, vosNum )
```

RETURN VALUES: Returns the number of sweeps in the volume scan or TK_FAIL if unsuccessful

PREREQUISITES: The volume scan corresponding to the granuleHandle must have been previously opened by TKopen().

NAME: TKsetL1GVtemplate()

DESCRIPTION: This routine creates a template for a GV product which is initialized with dimensions of the data product being written. This routine is used only for writing GV product files.

USAGE:
```
#include "IO.h"
#include "IO_GV.h"
INTEGER FUNCTION TKsetL1GVtemplate( nvos,
nparm(MAX_VOS), ncell(MAX_PARM,MAX_VOS),
nray(MAX_VOS), nsweep(MAX_VOS), fileName)
INTEGER  nvos
INTEGER  nparam(MAX_VOS)
INTEGER  ncell(MAX_PARM,MAX_VOS)
INTEGER  nray(MAX_VOS)
INTEGER  nsweep(MAX_VOS)
CHARACTER*TK_FNAME_LEN  fileName
```

INPUTS: nvos - Number of volume scans

nparam - Number of parameters for each volume scan

ncell - Array specifying the number of cells for each combination of volume scan and parameter.

nray - Number of rays for each volume scan

nsweep - Array specifying the number of sweeps for each volume scan.

fileName - Name of the HDF file in which L1 GV data will be stored

OUTPUTS: None.

DETAILS: This routine must be called prior to calling TKopen when writing a L1 GV data product. It creates a template that is used in dimensioning the HDF file that will be written. The first time the routine is called, it creates the template file. Subsequent calls append new templates to the existing file.

A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

EXAMPLES:

```
C       IO.h contains the function prototypes for the
C       majority of the I/O routines, including the GV
C       read and write routines.

        #include "IO.h"

C       Include the GV structure definitions and macros
        #include "IO_GV.h"

        RECORD /L1B_1C_GV/              l1BGVData
        RECORD /WRAPPER_HANDLE/         granuleHandle1BGV
        CHARACTER*TK_FNAME_LEN granuleID
        INTEGER  dataType
        CHARACTER                       filemode
        INTEGER  status
        INTEGER  nvos
        INTEGER  nparm(MAX_VOS)
        INTEGER  ncell(MAX_PARM,MAX_VOS)
        INTEGER  nray(MAX_VOS)
        INTEGER  nsweep(MAX_VOS)

C       Initialize the arrays
        DATA nparm /10*2/
        DATA ncell /20*230/
        DATA nray  /10*375/
        DATA nsweep /10*14/

C       Create a L1 GV template.  First set the values of
C       the GV file, then call the routine to create the
C       template file.

        nvos = 10
        granuleID = 'L1B_GV_input.dat'
        status = TKsetL1GVtemplate(nvos, nparm, ncell, nray,
                 nsweep, granuleID)

C       Check the error status
        IF (status .NE. TK_SUCCESS ) THEN
C         Perform error handling here
```

```
C       Open the file for writing */
        dataType = TK_L1B_GV
        filemode = TK_NEW_FILE
        status = TKopen(granuleID, dataType, filemode,
                granuleHandle1BGV)

C       Check the Error Status
        IF (status .NE. TK_SUCCESS) THEN
C       Perform error handling here
```

RETURN VALUES: TK_SUCCESS - If routine was successful TK_FAIL - If routine was unsuccessful

PREREQUISITES: This routine must be called prior to calling TKopen to open a GV granule for writing.

NAME: TKreadL1GV()

DESCRIPTION: This routine reads a L1 GV data product.

USAGE:
```
#include "IO.H"
#include "IO_GV.h"
INTEGER FUNCTION TKreadL1GV(granuleHandle, sGV )
RECORD /WRAPPER_HANDLE/ granuleHandle
RECORD /data type/ sGV
```

INPUTS: granuleHandle -
A structure containing information about the data product being read. granuleHandle is returned by TKopen().

sGV -
A structure containing one volume scan which is read from an HDF file. This structure must be delcared to correspond to the data file being read.

OUTPUTS: None

DETAILS: A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

EXAMPLES:
```
      C     IO.h contains the function prototypes for the
      C     majority of the I/O routines, including the GV
      C     read and write routines.

            #include "IO.h"

      C     Include the GV structure definitions and macros
            #include "IO_GV.h"

            RECORD /L1B_1C_GV/      l1BGVData
            RECORD /WRAPPER_HANDLE/     granuleHandle1BGV
            CHARACTER*TK_FNAME_LEN  granuleID
            INTEGER  dataType
            CHARACTER*1         filemode
            INTEGER  status
```

```
C       Open the file for reading
        granuleID = 'L1B_GV_input.dat'
        dataType = TK_L1B_GV
        filemode = TK_READ_ONLY
        status = TKopen(granuleID, dataType, filemode,
             granuleHandle1BGV)

C       Check the Error Status
        IF (status .NE. TK_SUCCESS) THEN
C       Perform error handling here

        status = TKreadL1GV(granuleHandle1BGV, l1BGVData )

C       Check the Error status
        IF (status .NE. TK_SUCCESS) THEN
C       Perform error handling here
```

RETURN VALUES: TK_SUCCESS if the routine is successful. If an error occurs, TKreportError is called and processing is terminated.

PREREQUISITES: The file corresponding to granuleHandle must have been previously opened by TKopen.

NAME: TKwriteL1GV()

DESCRIPTION: This routine writes L1 GV granules to a product data file.

USAGE: #include "IO.h"

```
INTEGER FUNCTION TKwriteL1GV(granuleHandle, sGV)
RECORD /WRAPPER_HANDLE/ granuleHandle
RECORD /data type/ sGV
```

INPUTS: granuleHandle -
A structure containing information about the file to which data can be written. granuleHandle is returned by TKopen().

OUTPUTS: sGV -
A structure containing one volume scan which is to be written to an HDF data product. This structure must be delcared to correspond to the data product being written.

DETAILS: A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

EXAMPLES:

```
      C     IO.h contains the function prototypes for the
      C     majority of the I/O routines, including the GV
      C     read and write routines.

            #include "IO.h"

      C     Include the GV structure definitions and macros
            #include "IO_GV.h"

            RECORD /L1B_1C_GV/              l1BGVData
            RECORD /WRAPPER_HANDLE/             granuleHandle1BGV
            CHARACTER*TK_FNAME_LEN granuleID
            INTEGER                         dataType
            CHARACTER*1                     filemode
            INTEGER                         status
            INTEGER  nvos
            INTEGER*4                       nparm(MAX_VOS)
            INTEGER*4                       ncell(MAX_PARM,MAX_VOS)
            INTEGER*4                       nray(MAX_VOS)
            INTEGER*4                       nsweep(MAX_VOS)
```

```
C       Initialize the arrays
        DATA nparm /10*2/
        DATA ncell /20*230/
        DATA nray  /10*375/
        DATA nsweep /10*14/

C       Create a L1 GV template.  First set the values of
C       the GV file, then call the routine to create the
C       template file.

        nvos = 10
        granuleID = 'L1B_GV_input.dat'
        status = TKsetL1GVtemplate(nvos, nparm, ncell, nray,
             nsweep, granuleID)

C       Check the error status
        IF (status .NE. TK_SUCCESS ) THEN
C       Perform error handling here
C       Open the file for reading
        dataType = TK_L1B_GV
        filemode = TK_NEW_FILE
        status = TKopen(granuleID, dataType, filemode,
             granuleHandle1BGV)

C       Check the Error Status
        if (status != TK_SUCCESS)
C       Perform error handling here

        status = TKwriteL1GV( granuleHandle1BGV, l1BGVData)
C       Check the Error status
        IF (status .NE. TK_SUCCESS) THEN
C       perform error handling here
```

RETURN VALUES:

TK_SUCCESS	- Successful reading of data
TK_FAIL	- Routine failed

PREREQUISITES: To use TKwriteL1GV, the user must first call TKsetL1GVtemplate(), and then call TKopen(). There must be a separate call to TKsetL1GVtemplate() (and a corresponding call to TKopen) for each file being written with TKwriteL1GV().

NAME: TKendOfFile()

DESCRIPTION: This routine determines the number of records in an HDF file and returns TK_EOF when an end of file is reached.

USAGE:

```
#include "IO.h"

INTEGER FUNCTION TKendOfFile( granuleHandle)
RECORD /WRAPPER_HANDLE/ granuleHandle
```

INPUTS: granuleHandle -
A structure containing information about the file to which data can be written. granuleHandle is returned by TKopen ().

OUTPUTS: None.

DETAILS: The first time this routine is called it determines the number of records in the HDF file. Each subsequent call to this routine decrements the number of records by one. When the number of records is zero, the routine returns TK_EOF, otherwise it returns TK_FAIL.

EXAMPLES:

```
C       IO.h contains the function prototypes for the
C       majority of the I/O routines, including the GV
C       read and write routines.

        #include "IO.h"

C       Include the I/O header file for TMI
        #include "IO_TMI.h"

        RECORD /L1B_11_SCANDATA/        l1B11Data
        RECORD /WRAPPER_HANDLE/             granuleHandle1B11
        CHARACTER*TK_FNAME_LEN          granuleID
        INTEGER                         dataType
        CHARACTER*1                     filemode
        INTEGER                         status

C       Open the file for reading
        granuleID = 'L1B_11_input.dat'
        dataType = TK_L1B_11
        filemode = TK_READ_ONLY
        status = TKopen(granuleID, dataType, filemode,
             granuleHandle1B11)
```

```
C       Check the Error Status
        IF (status .NE. TK_SUCCESS) THEN
C         Perform error handling here

        while (TKendOfFile( granuleHandle1B11 ) .NE.  TK_EOF)
        status = TKreadScan(granuleHandle1B11, l1B11Data)

C       Check the Error Status */
        IF (status .NE. TK_SUCCESS) THEN
C       perform error handling here

        ...
        endwhile
```

RETURN VALUES: TK_EOF - Indicates an end of file has been reached.
TK_FAIL - End of File conditions not encountered

PREREQUISITES: The file corresponding to granuleHandle must have been previously opened for reading by TKopen().

NAME: TKreadHeader()

DESCRIPTION: This routine reads PR ray header data from a 1B21, 1C21 data product, or reads clutter flags from a 2A25 product data file.

USAGE:
```
#include "IO.h"
INTEGER*2 TKreadHeader(granuleHandle, sHeader)
RECORD /WRAPPER_HANDLE/ granuleHandle
RECORD /L1B21_L1C21_HEADER/ sHeader
or
RECORD /CLUTTER_FLAGS/ sHeader
```

INPUTS: granuleHandle -
A structure containing information about the file from which data can be obtained. granuleHandle is returned by TKopen().

OUTPUTS: sHeader -
A structure containing the PR calibration coefficients and ray header for 1B21 and 1C21 data products, or the clutter flags for the 2A25 data product.

DETAILS: A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

EXAMPLES:
```
#include "IO.h"
C       Include the I/O header file for PR
        #include "IO_PR.h"

C       Declare Variables
        RECORD/L1B21_L1C21_HEADER/L1b21header
        RECORD/WRAPPER_HANDLE/granuleHandle1B21
        CHARACTER*TK_FNAME_LEN  granuleID
        INTEGER                 dataType
        CHARACTER*1             filemode
        INTEGER*2   status

C       Access the file for reading
        granuleID = L1B_21_input.dat
        dataType = TK_L1B_21
        filemode = TK_READ_ONLY
        status = TKopen(granuleID, dataType, filemode, granuleHandle1B21)
```

```
C       Check the Error Status
        if (status .NE. TK_SUCCESS)
C       handle error processing here

        status = TKreadHeader(granuleHandle1b21, L1b21header)

        Check the Error Status
        if (status != TK_SUCCESS)
C       handle error processing here
```

RETURN VALUES: TK_SUCCESS - Routine completed successfully.
W_TK_BADPIDRH - An invalid product ID was passed to TKreadHeader
W_TK_BADFILEMODRH - An invalid file mode was passed to TKreadHeader

PREREQUISITES: Before calling TKreadHeader(), a file must be opened for reading by calling TKopen(). When the file is no longer needed, it should be closed by calling TKclose().

NAME: TKwriteHeader()

DESCRIPTION: This routine writes PR ray header data to a 1B21, 1C21 data product, or writes clutter flags to a 2A25 product data file.

USAGE:
```
#include "IO.h"
INTEGER*2 TKwriteHeader(granuleHandle, sHeader)
RECORD /WRAPPER_HANDLE/ granuleHandle
RECORD /L1B21_L1C21_HEADER/ sHeader
or
RECORD /CLUTTER_FLAGS/ sHeader
```

INPUTS: granuleHandle -
A structure containing information about the file from which data can be obtained. granuleHandle is returned by TKopen().

OUTPUTS: sHeader -
A structure containing the PR calibration coefficients and ray header for 1B21 and 1C21 data products, or the clutter flags for the 2A25 data product.

DETAILS: A detailed description of the input and output parameters and return codes can be found in the appendix "Parameter Dictionary".

EXAMPLES:
```
#include "IO.h"
```
C
```
Include the I/O header file for PR
#include "IO_PR.h"
```

C
```
Declare Variables
RECORD/L1B21_L1C21_HEADER/ L1b21header
RECORD/WRAPPER_HANDLE/ granuleHandle1B21
CHARACTER*TK_FNAME_LEN granuleID
INTEGER         dataType
CHARACTER*1     filemode
INTEGER*2       status
```

C
```
Access the file for writing
granuleID = L1B_21_input.dat
dataType = TK_L1B_21
filemode = TK_NEW_FILE
status = TKopen(granuleID, dataType, filemode, granuleHandle1B21)
```

```
C       Check the Error Status
        if (status .NE. TK_SUCCESS)
C       handle error processing here

        status = TKreadHeader(granuleHandle1b21, L1b21header)

        Check the Error Status
        if (status != TK_SUCCESS)
C       handle error processing here
```

RETURN VALUES: TK_SUCCESS - Routine completed successfully.
W_TK_BADPIDWH - An invalid product ID was passed to TKwriteHeader
W_TK_BADFILEMODWH - An invalid file mode was passed to TKwriteHeader

PREREQUISITES: Before calling TKreadHeader(), a file must be opened for reading by calling TKopen(). When the file is no longer needed, it should be closed by calling TKclose().